University of Alberta

Reliability Evaluation and Optimal Design of Multi-State Weighted Systems

by

Wei Li ©

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Engineering Management

Department of Mechanical Engineering

Edmonton, Alberta
Fall 2008

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-46360-4
Our file Notre référence
ISBN: 978-0-494-46360-4

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

This thesis is dedicated with love and respect to

my father, mother and my beloved family

ABSTRACT

Reliability is one of the most critical performance measures of modern systems, and has been emphasized more and more by academia, industry and government. In traditional binary reliability frameworks, both systems and components can take only two possible states: completely working and totally failed. However, engineering systems typically have multiple partial failure states in addition to the above-mentioned two states. Reliability analysis considering multiple possible states is known as multi-state reliability analysis. Furthermore when a multi-state component or system is in a certain state, it can produce a certain amount of benefit. Such benefit is called the component or system utility. When both component and system utility are considered in a multi-state system model, the model is called a multi-state weighted system model. When a multi-state system model is extended into a multi-state weighted system model, it becomes more effective and flexible.

Efficient reliability evaluation and optimal design methods have not always been available for multi-state weighted systems. Without such methods, it is time-consuming, and sometimes impossible, to evaluate the reliability of complex systems and find the optimal solutions of reliability based design. Approximation approaches have had to be used. This dissertation documents research contributions to multi-state weighted system reliability theory, including reliability modeling, evaluation and optimal design.

The contributions are summarized as follows:

(1) Brought forward the definitions of and developed efficient algorithms for the reliability evaluation of multi-state weighted systems with *k*-out-of-*n* structures and series-parallel structures.

(2) Optimal reliability design based on component selection has been studied by two approaches for the binary weighted *k*-out-of-*n* systems which is a special case of multi-state weighted *k*-out-of-*n* systems.

(3) The more general situation – optimal reliability design based on component design has been studied for both multi-state weighted *k*-out-of-*n* systems and multi-state weighted series-parallel systems.

(4) An effective method has been developed for dealing with multiple objectives optimization typically involved in the optimal reliability design of multi-state weighted series-parallel systems.

With efficient reliability evaluation methods and effective optimal reliability design approaches for multi-state weighted engineering systems, my research results provide useful tools for achieving highly reliable and cost-effective engineering systems.

ACKNOWLEDGEMENTS

I would like to express my gratitude and sincere thanks to my supervisor Dr. Ming J. Zuo for his invaluable support, guidance and encouragement through all of my study and research.

I also would like to thank my Ph.D. examining committee members, Dr. John Doucette, Dr. Ted R. Heidrick, Dr. Don Koval and Dr. David W. Coit, for their help in the development of this thesis. Thanks to Dr. Coit in particular for taking the efforts to travel from Rutgers University, U.S. to attend my final oral examination. And thanks to Dr. Michael G. Lipsett for chairing my final oral examination.

Special thanks to Dr. Doucette for his kind help in my research and career development. I am also grateful to all the members in our Reliability Research Lab and the Advanced Structures and Materials Lab for their help and support.

TABLE OF CONTENTS

1. INTRODUCTION 1

1.1 Reliability 1

1.2 Multi-state weighted systems reliability and several practical examples 3

1.3 The research topics 8

1.4 Organization of the thesis 9

2. LITERATURE REVIEW OF BINARY WEIGHTED SYSTEM RELIABILITY AND MULTI-STATE SYSTEM RELIABILITY 10

2.1 The introduction of binary weighted system reliability and multi-state system reliability 10

2.2 Binary weighted system reliability evaluation 12

2.3 Multi-state system reliability evaluation 12

2.4 Optimal design of multi-state systems 18

3. FUNDAMENTALS OF MULTI-STATE WEIGHTED SYSTEM RELIABILITY 41

3.1 Basic concepts 41

3.2 Typical system structures 42

3.2.1 Series-parallel systems 43

3.2.2 *K*-out-of-*n* systems 45

3.3 Recursive Algorithms 46

3.4 Universal Generating Functions 48

4. RELIABILITY EVALUATION OF MULTI-STATE WEIGHTED *K*-OUT-OF-*N* SYSTEMS 52

4.1 Binary weighted *k*-out-of-*n*:G systems 52

4.2 Multi-state weighted *k*-out-of-*n*:G system: model I 58

4.3 Multi-state weighted *k*-out-of-*n*:G system: model II 66

4.4 Conclusions 73

5. OPTIMAL DESIGN OF BINARY WEIGHTED *K*-OUT-OF-*N* SYSTEMS 76

5.1 Introduction 77

5.2 Design models 79

5.3 Solution approaches and example results 83

5.4 Conclusions 96

6. OPTIMAL DESIGN OF MULTI-STATE WEIGHTED *K*-OUT-OF-*N* SYSTEMS BASED ON COMPONENT DESIGN 102

6.1 Introduction 103

6.1.1 The binary weighted *k*-out-of-*n* system model 103

6.1.2 The multi-state weighted *k*-out-of-*n* system models 104

6.2 Design models 107

6.3 Solution approach and an illustration example 122

6.4 Conclusions 126

7. OPTIMAL DESIGN OF MULTI-STATE WEIGHTED SERIES-PARALLEL SYSTEM USING PHYSICAL PROGRAMMING AND GENETIC ALGORITHMS 131

7.1 Introduction 132

7.2 Multi-state weighted series-parallel system 136

7.2.1 The definitions of MS weighted series-parallel systems 139

7.2.2 Reliability evaluation of MS weighted series-parallel systems 147

7.3 Design model 149

7.4 Solution approach and an illustration example 156

7.5 An alternative model of the system 159

7.6 Conclusions 164

8. CONCLUSIONS AND FUTURE WORK 170

8.1 Conclusions 170

8.2 Future work 173

LIST OF TABLES

Table 1.1: Physical Condition Assessment 4

Table 4.1: CPU time comparison of the Wu and Chen algorithm and the UGF approach as a function of system size n (k=200) 57

Table 4.2: Reliability distribution of the components, $p_{i,j}$ 60

Table 4.3: Utility distribution of the components, $w_{i,j}$ 60

Table 4.4: Comparison of the CPU times of the two methods when only M changes with random integer number for the component utility ($k_1 = 200$, $n = 5$) 65

Table 4.5: Comparison of the CPU times of the two methods when only n changes with random integer number for the component utility ($k_1 = 200$, $M = 5$) 65

Table 4.6: Comparison of the CPU times of the two methods when only n changes with random decimal number for the component utility ($k_1 = 200$, $M = 5$) ... 65

Table 4.7: Comparison of the CPU times of the two methods when only M changes with random decimal number for the component utility ($k_1 = 200$, $n = 5$) 66

Table 4.8: Comparison of the CPU times of the two methods when only M changes with random integer number for the component utility ($k_1 = 200$, $n = 5$) 71

Table 4.9: Comparison of the CPU times of the two methods when only n changes with random integer number for the component utility ($k_1 = 200$, $M = 5$) 71

Table 4.10: Comparison of the CPU times of the two methods when only M changes with random decimal number for the component utility ($k_1 = 200$, $n = 5$) 72

Table 4.11: Comparison of the CPU times of the two methods when only n changes with decimal decimal number for the component utility ($k_1 = 200, M = 5$) 72

Table 4.12: Comparison of system state distributions of Model I and Model II 72

Table 5.1: Characteristics of available components: Database 1 89

Table 5.2: All possible type 1 moves from the initial solution 91

Table 5.3: All possible type 2 moves from the initial solution 91

Table 5.4: Comparison of GA and TS on CPU time for Problem **P1** based on database 1 94

Table 5.5: Comparison of GA and TS on CPU time for problem **P2** based on database 1 95

Table 6.1: The price for the Western Digital Hard Disks (Ultra ATA-100 7200-RPM 8MB 3.5 IDE HDD) (www.wdc.com) 113

Table 6.2: The price for the Archos MP3 player Compact Flash Memory Card (www.18004memory.com) 113

Table 6.3: Parameters used in the example 124

Table 6.4: Optimization results in Model I 125

Table 6.5: Optimization results in Model II 125

Table 7.1: State distribution of component 1 139

Table 7.2: State distribution of component 2 139

Table 7.3: Utility distribution of the MS weighted series system 140

Table 7.4: Utility distribution of the MS weighted parallel system 141

Table 7.5: State distribution of component 3 143

Table 7.6: State distribution of component 4 144

Table 7.7: Utility distribution of the MS weighted series-parallel system in state 0 144

Table 7.8: Utility distribution of the MS weighted series-parallel system in state 1 145

Table 7.9: Utility distribution of the MS weighted series-parallel system in state 2 145

Table 7.10: State distribution of the MS series-parallel system 146

Table 7.11: Parameters used in the component cost function 156

Table 7.12: Physical programming class functions setting 157

Table 7.13: Comparison of the single-objective optimization results and multi-objective optimization results for multi-state weighted serials-parallel. 158

Table 7.14: Comparison of the single-objective optimization results and multi-objective optimization results for the alternative multi-state weighted serials-parallel system 163

LIST OF FIGURES

Figure 1.1: A power generation unit of the Israel Electric Corporation Ltd. 5

Figure 1.2: Distribution of all events for a 360MW coal-fired unit at Israel Electric Corporation Ltd. 5

Figure 1.3: Grouping into bins of all events for a 360MW coal-fired unit at Israel Electric Corporation Ltd. 6

Figure 1.4: State space diagram for a 360MW coal-fired unit at Israel Electric Corporation Ltd. 7

Figure 3.1: Structure of a multi-state weighted series-parallel system 44

Figure 5.1: Type 1 move 86

Figure 5.2: Type 2 move 86

Figure 5.3: The flow chart of the tabu search procedure 88

Figure 5.4: Comparison of GA and TS on CPU time for Problem **P1** based on database 1 95

Figure 5.5: Comparison of GA and TS on CPU time for problem **P2** based on database 1 96

Figure 6.1: Component cost as a function of reliability and utility 116

Figure 7.1: Structure of a multi-state weighted series-parallel system 138

Figure 7.2: An example of MS weighted series-parallel system 143

Figure 7.3: Qualitative meaning of soft class function 152

Chapter 1
Introduction

Reliability engineering is the discipline of ensuring that a system will be reliable when operated in a specified manner. Reliability engineering work is performed throughout the entire life cycle of a system, including development, testing, production and operation. Growing international competition has increased the need for all designers, managers, practitioners, scientists and engineers to ensure the highest possible level of reliability for their products within the constraints of budget and other resources. As modern systems are becoming more and more complex, research on effective approaches to measuring their reliability and developing optimal reliability designs becomes more and more important.

1.1 Reliability

Reliability is a broad term that focuses on the ability of a product to perform its intended function. Mathematically speaking, assuming that an item is performing its intended function at time equals zero, reliability can be defined as the probability that an item will continue to perform its intended operating function without failure for a specified period of time under stated operating conditions. The product defined here could be an electronic or mechanical hardware product, a software product, a manufacturing process, or even a service. Reliability engineers perform a wide variety of special management and engineering tasks to ensure that sufficient attention is given to reliability tasks. The production of a particular system includes the following specific reliability tasks [1]:

- Systematically collecting lifetime data of a device for reliability evaluation from the field or tests.
- Reliability modeling for practical systems, and analyzing system reliability based on the reliability of components.
- Enhancing system reliability through optimal design and maintenance scheduling, followed by verifying decisions by thorough analysis and testing.
- Evaluating the reliability potential of alternative designs.

The reliability of a system needs to be evaluated accurately, and it can be improved through design optimization. In traditional binary reliability frameworks, both systems and components can take only two possible states: perfectly working and totally failed [1]. Traditionally, reliability is the probability of a component/system to be at the perfect working state. However, typically in many real-life situations, systems and their components are capable of assuming a whole range of levels of performance degradation, varying from perfect functioning to complete failure. The dichotomous model is an oversimplification of actual systems. In today's real world problems, the great number of system states needs to be considered, and increasingly high requirements for accurate reliability evaluation and optimal design make it difficult to use traditional binary reliability techniques. Therefore, the reliability theory of multi-state (MS) reliability is highly needed.

1.2 Multi-state weighted systems reliability and several practical examples

Reliability analysis considering multiple possible states is known as multi-state (MS) reliability analysis. MS reliability analysis recognizes the multiple possible states of engineering systems, and enables more accurate system reliability analysis and design.

In a multi-state system (MSS) reliability model, the system and each component may experience states in the set $\{0, 1, 2, ..., M\}$. Furthermore when a multi-state component or system is in a certain state, j, it can produce a certain amount of benefit. Such benefit is called the component or system utility. When the utilities of the component and system are considered in the MSS model, the model is a multi-state weighted system model. Of course, the binary weighted system is a special case of MS weighted systems in which there are only two states of the component and the system: working or failed. Two practical examples of MS weighted systems are given below.

Example 1.1: Highway infrastructure system [1]

In the City of Edmonton, highway infrastructure deterioration conditions are evaluated based on the scale shown in Table 1.1. The highway infrastructure system has five states (A-E). The physical condition of highways in each state is the parameter that describes the performance capacity of the highways. In this thesis, we call it the utility. Each highway follows a certain deterioration behavior from state A to state E. With certain rehabilitation strategies, the highway health condition can be improved at a certain cost.

Table 1.1: Physical Condition Assessment [1]

Mark	State	Explanation of condition
A	Very good or Performance more than 90%	Element is structurally sound and is functional as intended when it was designed. Maintenance and operations costs are well within standards and norms. Element is new or recently undergone major rehabilitation. Its condition and function are practically equal to a new.
B	Good or Performance more than 70%	Element is structurally sound and is functional as intended when it was designed. Maintenance and operations costs are within acceptable levels but increasing with time. Typically such infrastructure would have not reached its midlife span.
C	Fair or Performance more than 60%	Element is showing signs of aging, small portions can be structurally deficient or the element is becoming functionally obsolete. Such an element is approaching the stage where expenditures beyond the original planned maintenance are being incurred to keep it useable.
D	Poor or Performance more than 40%	Element is approaching a poor condition. Signs of structural deficiency are becoming more pronounced and obvious. The element's physical condition may be contributing to safety hazards or negatively impacting safety, health, environment, or other areas.
E	Inadequate or Performance less than 40%	Element is structurally unsound and/or is not functional anymore. It is only a matter of time before it completely fail.

Example 1.2: Power generation units [2]

Power generation units are typical MS weighted systems. A power generation unit as in Figure 1.1 can perform not only at its full capacity level, but also at other lower capacity levels based on its health condition. Here the different states are based on the different performance capacities of the power unit, not based on the demand for the unit. During a ten year period of time, the different output capacity levels of a 360MW (full capacity) coal-fired power generation unit at Israel Electric Corporation Ltd. were illustrated in Figure 1.2. The coal fired power generation unit experienced 36 different states during

the ten year period. Different states have different utilities which are the output capacities corresponding to them.

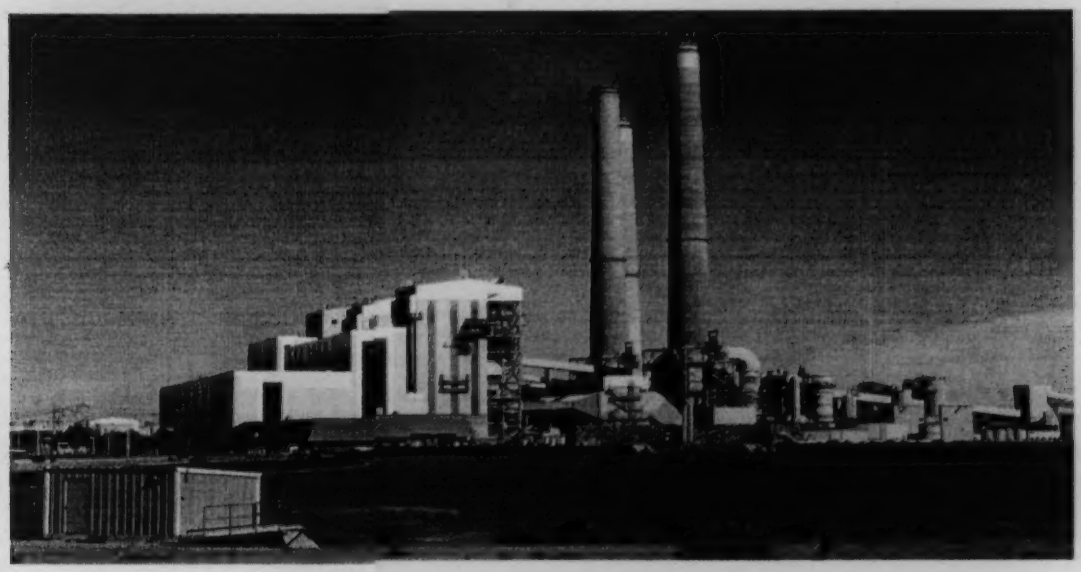


Figure 1.1: A power generation unit of the Israel Electric Corporation Ltd. [2]

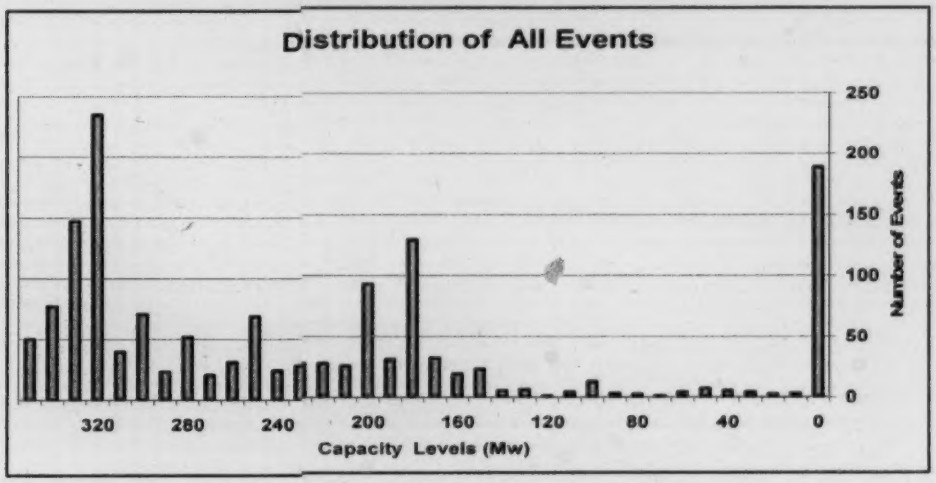


Figure 1.2: Distribution of all events for a 360MW coal-fired unit at Israel Electric Corporation Ltd. [2]

The distribution of all failure events was classified into 9 states by bubble type bins in Figure 1.3. Using the average capacity in each bubble as the capacity in that state, together with the full capacity state, ten states with their corresponding capacities are illustrated in Figure 1.4. During the operation, the coal-fired generation unit will deteriorate from the full capacity state to other lower states. Not all the state transitions are shown here. Transactions between the degraded states are not shown in this figure. With the help of maintenance activities, the unit can be rehabilitated from a lower state to the full capacity state. Figure 1.4 is used to describe some state transitions in this MS weighted system. Although repair activities are illustrated in this figure, it will not be studied in this thesis. In this thesis, we only consider non-repairable systems.

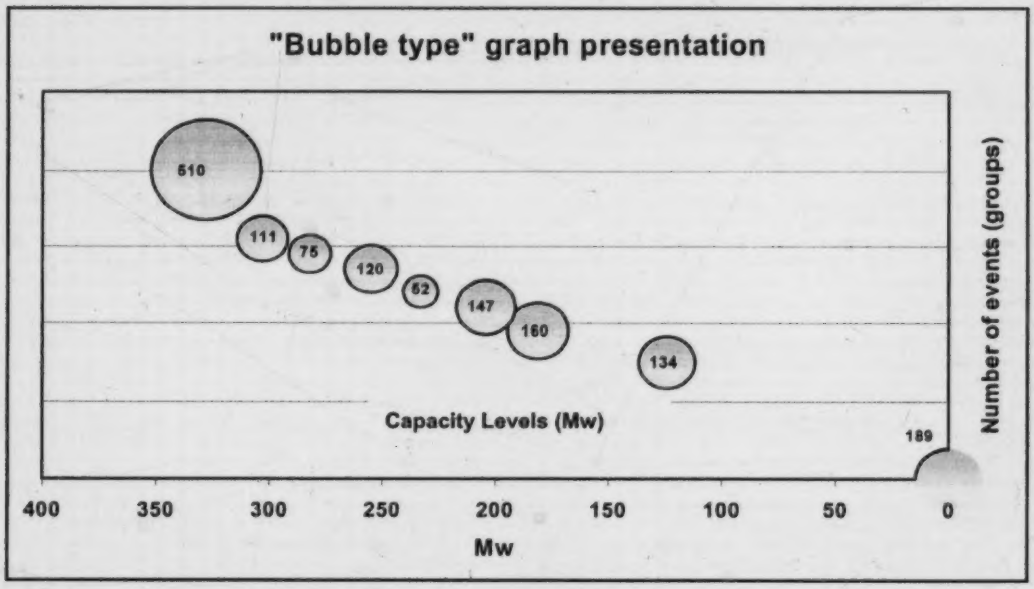


Figure 1.3: Grouping into bins of all events for a 360MW coal-fired unit at Israel Electric Corporation Ltd. [2]

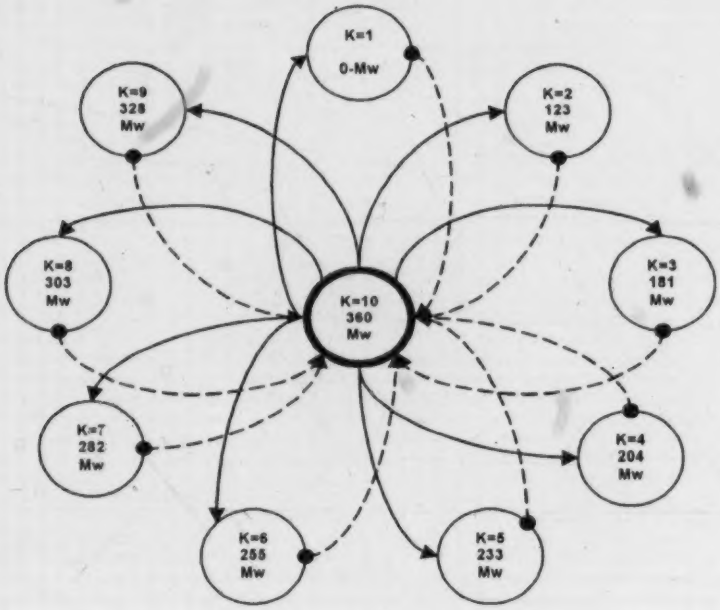


Figure 1.4: State space diagram for a 360MW coal-fired unit at Israel Electric Corporation Ltd. [2]

In the multi-state system, both the component and the system have more than two states. There is a probability value corresponding to each state. Generally, a pre-specified utility value is used to describe the system performance level in each state. In the literature, most of the research problems in MS system reliability have focused on how to evaluate the probability of the system being in each state based on the state probability distribution of components, and how to maximize system reliability subject to given resources or vice versa. A detailed literature review of MS reliability will be given in Chapter 2. In this thesis, a more general model MS weighted system model is brought forward and studied for different practical structures. Both the reliability evaluation and the optimal reliability design are studied for the MS weighted system.

1.3 The research topics

As mentioned in Section 1.1, system reliability evaluation and enhancing system reliability through optimal design are two very important tasks in reliability engineering. However, because the traditional binary reliability system model is an over-simplified model of the real-world situation, the system reliability evaluation and optimal design based on binary system models are not accurate enough. In addition, although the MS system reliability model has been brought forward and studied with regard to reliability evaluation and optimal reliability design, its modeling ability is limited. In the traditional MS system model, because there is no parameter to describe the component (system) performance capacity, we have to assume that the people using the model know the component (system) performance capacity corresponding to each state. However, this may not always be true. In this thesis, a more general MS system model — the MS weighted system model is brought forward. In this model, we use a parameter - component (system) utility to describe the component (system) performance capacity, a system utility distribution is calculated based on the component utility distribution. When the MS system model is extended to the MS weighted system model, the model becomes more effective and flexible.

In this thesis, two different MS weighted system models (MS weighted k-out-of-n systems and MS weighted series-parallel systems) will be brought forward and their reliability evaluation and optimal design problems will be studied.

1.4 Organization of the thesis

The thesis, consisting of 8 chapters, is organized as follows. Chapter 2 is the literature review of multi-state system reliability. Chapter 3 gives the fundamental knowledge of multi-state weighted reliability, including basic concepts and typical multi-state weighted system structures. Chapters 4, 5, 6 and 7 discussed the reliability evaluation and optimal reliability design of three typical weighted systems: binary weighted *k*-out-of-*n* systems, multi-state weighted *k*-out-of-*n* systems and multi-state weighted series-parallel systems.

The contributions of this research work are summarized in Chapter 8, and some future work is discussed.

References:

[1]. H. AL-Battaineh. Developing an Optimal Best Practice Guidelines for Infrastructure Systems. PhD thesis, University of Alberta, 2007.

[2]. Y. Ding. Multi-State Systems Reliability Modeling and Evaluation. Presentation, Reliability Research Lab, Mechanical Engineering Department, University of Alberta, 2007.

[3]. P.D.T. O'Connor. Practical Reliability Engineering, John Wiley & Sons, 2002.

Chapter 2

Literature Review of Binary Weighted System Reliability and Multi-state System Reliability

This chapter gives a literature review of the research related to MS weighted system reliability. Since there are no published results specifically on MS weighted system reliability, the literature review focuses on binary weighted system reliability which is a special case of MS weighted system reliability and general MS system reliability. For the literature review on traditional binary system reliability, please refer to [41] and [83].

2.1 The introduction of binary weighted system reliability and multi-state system reliability

Although the topic of multi-state (MS) weighted system reliability is new, there are already some studies on binary weighted system reliability and MS system reliability.

In 1994, Wu and Chen [109] generalized the binary k-out-of-n system reliability model to the binary weighted k-out-of-n system reliability model. In 1968, Hirsch et al. [24] brought forward the basic idea of multi-state systems. In the 1970s and 1980s, more and more people joined the multi-state reliability research. Ross [90] studied a component or system with more than two possible states. Barlow and Wu [4], El-Neweihi et al. [16] and Natvig [75] studied coherent multi-state systems. Block and Savits [8] studied multi-state monotone systems. The results from the early studies on multi-state reliability were generalized in the work of Griffith [20], and Hudson and Kapur [33]. Their work studied the primary concepts of multi-state reliability, including system structure function, minimal cut (path) set, relevancy and coherency. The reliability importance was extended

to multi-state systems in [11], [20], [19] and [8]. The early advances in multi-state reliability theory were summarized by El-Neweihi and Proschan [17].

The commonly used binary reliability models — the series-parallel system, *k*-out-of-*n* system and network system — have all been extended from the binary context to the multi-state context. These systems will be introduced in detail in Chapter 3. The multi-state series-parallel system was studied in [4], [44] and [55]. The multi-state *k*-out-of-*n* system was studied in [9], [16], [29] and [32]. The multi-state network system was studied in three classes. In [62], [88], [89] and [93], both the links and the nodes can have more than 2 states. In [112] and [113] only the nodes can have more than 2 states, the links are perfect. In [61] and [121], only the links can have more than 2 states, the nodes are perfect. In addition, in the binary context, a consecutive *k*-out-of-*n* system is failed if and only if at least *k* consecutive components are failed. This model has also been extended to the multi-state context and studied in [31], [111] and [118].

There is more than one way to extend a binary reliability model to the multi-state model. For example, in Barlow and Wu's definition of multi-state series-parallel systems [4], the state of a parallel subsystem is equal to the state of the best component in that parallel subsystem. However, in Levitin's definition of multi-state series-parallel systems [55], the capacity of a parallel subsystem is equal to the sum of the capacities of its constituent components. Under the traditional definition of the multi-state *k*-out-of-*n*: G system, [9] and [16], the system is in state *j* or above when at least *k* components are in state *j* or above. Huang et al. [29] and [32] proposed the model of the generalized multi-state *k*-out-

of-*n*: G system by allowing different requirements for a number of components in different states.

2.2 Binary weighted system reliability evaluation

To evaluate the reliability of a binary weighted *k*-out-of-*n* system, Wu and Chen [109] provided a recursive algorithm. This algorithm will be introduced in details in Chapter 3. Higashiyama [23] provided a method to express the system reliability of a weighted *k*-out-of-*n* system in fewer terms than that obtained with the algorithm by Wu and Chen [109]. However, the time complexity and space complexity of these two methods reported in [23] and [109] are the same. The most recently reported study on binary weighted *k*-out-of-*n* system is by Chen and Yang [13]. In [13], the one-stage weighted *k*-out-of-*n* model was extended to the two-stage weighted *k*-out-of-*n* model with components in common among the stages. The one-stage weighted *k*-out-of-*n* system does not contain any subsystems. The two-stage weighted *k*-out-of-*n* system consists of a number of subsystems.

All the published results in binary weighted system reliability focus on reliability evaluation. This thesis will study the optimal reliability design of binary weighted *k*-out-of-*n* systems.

2.3 Multi-state system reliability evaluation

Successful reliability work must include the ability to model and evaluate system reliability. Many authors have made contributions which enrich multi-state system reliability modeling and evaluation theory.

Generally, MS system reliability evaluation methods are based on five different approaches: an extension of binary models to MS cases, the stochastic process approach, the universal generating function approach, the Monte-Carlo simulation technique and the recursive algorithm.

The approach based on the extension of Boolean models is historically the first approach that was developed and applied to MS system reliability evaluation. In 1980, Caldarola [12] assigned a Boolean variable to each component state in a MS system. This variable indicates whether the component is at this particular state or not. Since a number of Boolean variables may belong to the same component, the variables are no longer independent. A special technique called Boolean algebra with restrictions on the variables has been used to deal with this dependence. Based on this method, a MS system can be reduced to a binary system for which reliability evaluation methods such as Fault Trees have already been well developed. Aven [2] proposed an algorithm based on state-space decomposition. In [107], a factoring solution was also used to deal with the dependence among Boolean variables. Block Diagrams and Fault Trees were both applied to solving the reduced binary model. In order to deal with the great number of variables and the dependencies among the variables, an inclusion-exclusion formula was used in [106] to obtain the expressions for state probabilities. A Binary Decision Diagram (BDD) approach [114] has also been proposed to realize the Boolean algebra. BDD is a data structure that is used to represent a Boolean function. The nature of the BDD technique allows the sum of the disjoint products to be implicitly represented; this avoids using a great deal of storage space and reduces the computation burden. Applying the Boolean methods to determining the MS system structure and reliability measures in practical

situations can be found in [9]. The determination of the MS system structure function for applying the Boolean methods is not a trivial problem in many practical cases. Boedidheimer and Kapur [9] presented a methodology for customer-centered structure function development. According to this method, customers define the appropriate number of states for components and the entire MS system. The performance measures they developed relate the component states to the system states by specifying when a change in the state of any one of the components forces a change in the state of the whole system.

The stochastic process approach has been used to evaluate the reliability of MS systems and applied to power systems. In 1984, Natvig and Streller [76] first applied the stochastic process approach to MS system reliability evaluation. They analyzed the steady-state behavior of monotone MS systems by applying the stationary and synchronous stochastic process. Using stochastic process theory, Vaurio [105] developed models for the MS component reliability and availability evaluation with general failure and repair time distributions. The idea of combining the Markov processes and coherent structure function was proposed by Xue and Yang [110]. In [42], Lanus, Yin and Trivedi partitioned complex Markov models into a hierarchy of sub-models to obtain common availability measures for MS telecommunication systems. In [59], the components were checked periodically to obtain the state sequences of all components, and then this information was used to predict the reliability of the components in several periods, for example, the probability that the components are in specified states. Finally, the states of components in a number of periods were used by stochastic processes to calculate the

system reliability for those periods. MS system reliability evaluation was systematically studied using the stochastic processes in [3] and [10].

The universal generating function (UGF) was introduced by Ushakov [103] in 1986. In 1996, Lisnianski et al. [63] first applied UGF to power system reliability analysis. In a series of works, [44], [55], [52], [57], [62] and [63], different operators provided the determination of the entire MS system performance distribution based on the performance distributions of components. The system structures they studied include series, parallel, series-parallel and bridge structure. In of the some literature, UGF is also called the universal moment generating function (UMGF) [65], [66] and [73].

Technically, Monte-Carlo simulation can be used for the reliability evaluation of almost every real world MS system, although the main disadvantages of simulation are the time and expense involved in the development and execution of the models. In [7], a hybrid approach was presented using Monte-Carlo simulation and an enumeration technique for the reliability evaluation of large scale composite generation-transmission systems, including multi-state representation of generating units. Zio and Podofillini [116] have presented a Monte-Carlo simulation approach to estimate all the importance measures of the components at a given performance level in a multi-state series-parallel system, provided that the components are independent. In [117], the Monte-Carlo simulation method was used for modeling MS system reliability. The flexibility of the simulation method was exploited to study two models with dependant elements: parallel elements with load-sharing and parallel elements with operational dependencies. In [115], Zio, Marella and Podofillini continued their research on Monte-Carlo simulation modeling of

the complex dynamics of MS components subject to operational dependencies with the system overall state. Ramirez-Marquez and Coit [88] brought forward and compared a new Monte-Carlo simulation approach that obtained accurate approximations to actual MS two-terminal network system reliability. Previous exact calculation approaches have relied on enumeration or on the computation of multi-state minimal cut vectors and the application of inclusion/exclusion formulae. In [84], a multi-state Monte Carlo simulation model of a multi-state railway network system was developed.

Although use of a recursive algorithm is a new approach which has just been introduced into the MS system reliability field in the last several years, it has demonstrated its advantages in effectiveness and efficiency. Tian, Zuo and Yam [101] and Zuo and Tian [120] proposed a recursive algorithm for the reliability evaluation of generalized MS *k*-out-of-*n* systems defined by Huang et al [32]. Huang et al. [32] have provided an algorithm for calculating the state distribution of generalized multi-state *k*-out-of-*n*:G systems. However, their performance evaluation algorithm is enumerative in nature, and therefore is not efficient. Tian, Zuo and Yam [102] also developed a new general multi-state *k*-out-of-*n* system model which they can make more practical engineering systems fit into, and brought forward a new recursive algorithm for evaluating its reliability distribution. Tian and Zuo [97] continued their research in this field by proposing a so-called "unified MS *k*-out-of-*n* model". This model aims at providing high flexibility for modeling and analyzing practical and complex problems involving MS *k*-out-of-*n* structures. Efficient recursive algorithm for reliability evaluation of this model has also been developed [97]. Zuo and Tian [121] also studied the reliability evaluation of two-terminal MS networks using a recursive algorithm. They proposed a recursive algorithm

based on the SDP (sum of disjoint products) principle for the evaluation of MS network reliability, and confirmed its efficiency by comparing it with Aven's algorithm [2] which is recognized as efficient.

For complex systems with a large number of components and a large number of states, reliability bounds can assist us in efficient decision making. Minimal path vectors or minimal cut vectors not only can be used for exact reliability evaluation, but also for boundary evaluation. Several binary reliability bounding approaches are generalized to multi-state systems by Block and Savits [8], and analyzed by Meng [68]. These approaches use simple formulas generalized from the binary case. In the early 1980s, Hudson and Kapur [33], [34], [35], and [36] developed methods using minimal cut vectors for exact reliability evaluation and reliability bounding of multi-state series-parallel systems. In [93], Satitsatian and Kapur presented a new algorithm for finding lower boundary points, and used them to compute the reliability bounds for MS two-terminal networks. Huang et al. developed bounding approaches for generalized multi-state k-out-of-n:G systems [30] and consecutive multi-state k-out-of-n systems [31] by simplifying the minimal path or cut vectors to include no more than two different states. Tian, Yam and Zuo [98] proposed a systematic and flexible reliability bounding approach for MS k-out-of-n systems based on the recursive algorithm.

From the above, it is clear to see that in the research on MS system reliability evaluation, more and more general MS system models have been brought forward and more and more efficient evaluation approaches have been developed. Until now, the most efficient reliability evaluation method has been the recursive method. In this thesis, the most

general MS system model so far — MS weighted system model — will be brought forward and the recursive method will be developed to evaluate system reliability.

2.4 Optimal design of multi-state systems

An important direction for MS system reliability research is the development of optimization algorithms to solve different application problems. Two approaches have been used in the reliability optimization field. One aims at achieving the greatest possible reliability subject to different constraints. The other aims at minimizing the resources subject to a minimum reliability. Basically, there are four ways to enhance the reliability of multi-state systems [62]: (1) to provide redundancy, such as adding redundant components in parallel and using redundancy in the form of k-out-of-n systems (generally, all the components are assumed to be independent); (2) to adjust system configuration while keeping constituent components the same, such as optimal arrangement of the existing components; (3) to enhance the reliability or performance of the components; (4) a combination of the above three methods. Applied to the MS system, these methods affect two basic system properties: its configuration (structure function), and the performance distribution of its components. Therefore, all MS system optimal reliability design problems can be divided into two classes: structure optimization problems and MS components optimal reliability design problems. Most of the reported research has focused on the first problem until now.

The most popular MS system structure optimization problems studied in the literature are redundancy optimization problems. In order to solve practical problems in which a variety of products (versions of components) characterized by their performance,

reliability and price exist in the market, reliability engineers should have an optimization methodology for finding the optimal system structure by choosing the appropriate versions as well as a number of parallel components from the list of available products in the market. The reliability-redundancy allocation problem was first introduced by Misra and Ljubojevic [74] and has been intensively studied in the binary context [39]. The first redundancy optimization problem for multi-state systems was introduced in [103] where the general optimization approach was formulated. A modification of the gradient method was applied in [104] to finding the minimal cost configuration of a MS series–parallel power system structure. Components of the power system with different capacities and costs were considered, and the demand was estimated using a load curve. In the simplest MS series-parallel system structure optimization problem, each parallel subsystem can contain only identical elements [57]. The MS system structure optimization problem formulated in [57] was finding the minimal cost system configuration that provides the required level of the system performance measure. The more general situation in which different versions and the number of components may be chosen for any given parallel subsystem was studied in [58]. Genetic Algorithm (GA) was used to solve both the optimization problems in [57] and [58].

Ramirez-Marquez and Coit [87] studied the same optimization problem as in [58], but they used a heuristic algorithm instead of GA to solve the formulated optimization problem. The heuristic offered more efficient and straightforward analyses. This was the first time that the MS series-parallel system design problem has been addressed without using GA. The problem of structure optimization of MS controllable systems [46] is similar to the problem considered in [58]. The only difference is that the interaction

between the parallel subsystems is determined by the control rule that provides the maximal possible system performance in the MS controllable systems [46].

Instead of minimizing total investment cost subject to reliability constraints, Gupta and Agarwal [21] studied the optimization problem for MS series-parallel systems to maximize system reliability subject to system cost. They used GA together with the penalty technique to solve the formulated problem. Ant colony algorithm is another metaheuristics optimization technique used in MS systems structure optimization. In 2005, Massim, Zeblah et al. [66] showed how to use the ant colony algorithm to choose an optimal multi-state series–parallel power structure configuration in order to minimize total investment cost subject to availability constraints. They defined availability as the probability that a multi-state series–parallel system will be in a state having a capacity level greater than or equal to a specific demand at a specified moment. Meziane et al. [73] described how to implement the ant colony algorithm for finding the optimal multi-state series–parallel power system configurations needed to maximize system reliability subject to system performance and cost constraints. In 2007, Agarwal and Gupta [1] presented the ant colony algorithm for a homogeneous series-parallel system exhibiting a multi-state behavior, in order to minimize cost and provide a desired level of reliability. Liu, Zuo and Meng [64] formulated the optimization model for continuous multi-state series-parallel systems to determine the number of redundancies needed to maximize the system's expected utility function subject to constraints on system cost. The design goal is also achieved by discrete choices made from components available in the market. Wu and Chan [108] defined a new utility importance of a component state in MS systems and illustrated how genetic algorithm, simulated annealing, and tabu search can be used to

select components and define the position order of components so that the performance utility of a MS system is optimized. A summary of how mathematical programming, heuristics, metaheuristics, neural networks and fuzzy techniques have been used to solve formulated MS system structure optimization problems can be found in [43]. Besides the redundancy optimization problem, El-Neweihi, Proschan and Sethuraman [18] allocated MS components into k series systems in order to maximize the expected number of systems functioning at a pre-specified level or higher. Meng [67] extended the principle of interchanging components from binary systems to MS systems.

In practice, the designer often has to include additional elements in an existing system. It may be necessary, for example, to modernize a system according to new demand levels or new reliability requirements. The problem of minimal cost MS system expansion is very similar to the problem of system structure optimization in [57] and [58]. The only difference is that each MS parallel subsystem already contains some working elements. The problems of optimal single stage MS series-parallel system expansion was studied in [63]. The problem of optimal multi-stage MS series-parallel system expansion was studied in [48], [49] and [55]. The multi-stage expansion period is divided into several stages. At each stage, the demand distribution is predicted. Additional components chosen from the list of available products may be included in any parallel subsystem at any stage to increase total system reliability. The objective studied in the single stage [63], the multi-stage [48], [49] and [55] expansions was to minimize the sum of costs of investment over the study period while satisfying reliability constraints at each stage by adding additional components selected from a list of available products. Massim et al. [65] for the first time used the ant colony method to solve the multi-stage expansion problem

for multi-state series-parallel systems. The objective was to minimize the investment costs over the study period while satisfying availability or performance constraints by adding additional components selected from a list of available products.

Gurler and Kaya [22] proposed a maintenance policy for a system with multi-state components, using an approximation of the average cost function to reduce problem complexity, and using a numerical method to solve the formulated optimization problem. A heuristic algorithm was developed to solve the problem. Zuo et al. [119] investigated the replacement-repair policy for multi-state deteriorating products under warranty. When maintenance activities are considered together with construction optimization, the problem becomes more complicated. Levitin and Lisnianski [56] formulated the joint redundancy and maintenance replacement schedule optimization problem generalized to MS systems. Nourelfath and Dutuit [79] and Nourelfath and Ait-Kadi [78] extended the redundancy optimization problem of MS systems to the more general case where maintenance resources are limited. In [78] and [79], the classical redundancy optimization problem was extended to find, under reliability constraints, the minimal configuration and maintenance costs of a series–parallel system for which the number of maintenance teams is less than the number of repairable components. The difference between [78] and [79] is that they used different approaches to solve the formulated optimization problem. The authors of [79] used a method that is based on a combination of UMGF and a simulation method based on stochastic Petri nets animated by Monte-Carlo simulation. However, the approach developed in [78], using UMGF technique and the Markov chain approach, is mainly analytical. In [53] Levitin and Lisnianski summarized their technique for combining a UGF method used in reliability evaluation

and a GA used as an optimization engine for solving a family of MS system reliability optimization problems including structure optimization, optimal expansion, maintenance optimization and optimal multistage modernization

In practical situations involving reliability optimization, there often exist mutually conflicting goals such as maximizing system utility and minimizing system cost and weight [40]. Most reported multi-state optimization models such as those mentioned above treat one goal as the objective function and the other goals as constraints. However, it is very difficult to specify in advance the constraint values for the goals used as constraints. After a solution is obtained, we often need to modify these constraint values to find a better tradeoff between different goals. Finding the most appropriate constraint values is a trial and error process and there are no clear guidelines as to how to converge to the right set of constraint values. Particularly, when there are many constraint values that need to be specified, it is almost impossible to find the most appropriate values for these constraints. Compared to the traditional single-objective optimization model, the multi-objective optimization model which is used to seek the balance among several goals automatically is more flexible and effective. Although there has been intensive research on redundancy allocation for binary systems considering multiple objectives, it is a relatively new research topic in the multi-state context. For example, the studies of multi-objective optimization problems in the binary context can be found in [15], [25], [26], [27], [28], [37], [38], [81], [85], [86], [92] and [95]. The Fuzzy approach, crisp approach, genetic algorithm, interactive optimization method, and physical programming have all been used to build and solve multi-objective optimization problems. Physical programming has proved its effectiveness in addressing a wide array of multi-objective

optimization problems in the binary context [69], [70], [71], [72] and [82]. Tian, Zuo and Huang [100] and Tian and Zuo [96] studied the redundancy allocation problem for multi-state series-parallel systems using physical programming and the fuzzy approach. They compared the results from those two approaches and found that physical programming can generate better results.

Tian, Zuo and Huang [99] investigated how to improve the optimal design of multi-state series-parallel systems by extending the redundancy optimization of multi-state series-parallel systems to the joint reliability-redundancy optimization of multi-state series-parallel systems. That is, in addition to redundancies, component state distributions were treated as design variables as well. This is the only research in the literature on enhancing MS system reliability by modifying the performance distribution of its components.

Survivability, the ability of a system to tolerate intentional attacks or accidental failures or errors, becomes especially important when a system operates in battle conditions or is affected by a corrosive medium or other hostile environment. When applied to MS systems, the survivability depends on a MS system's ability to meet the demand (the required performance level). Subject to investment cost limitations, the optimization problems of how to find the minimal cost configuration of protection that provides the desired system survivability have been studied in [45], [46], [47], [48], [50], [51] and [54] for different types of MS systems.

Based on the research in the literature, a more general MS system model – the MS weighted system model – is brought forward in this thesis. And the research involves both reliability evaluation and optimal reliability design. Systems under investigation

include binary weighted *k*-out-of-*n* systems (a special case of MS weighted *k*-out-of-*n* systems), MS weighted *k*-out-of-*n* systems and MS weighted series-parallel systems. Specifically, in **Reliability Evaluation** of multi-state systems, we aim at developing efficient evaluation algorithms for MS weighted series-parallel systems and MS weighted *k*-out-of-*n* systems:

(1) MS weighted series-parallel systems

In order to model different practical situations, two MS weighted series-parallel system models are developed – the so-called MS weighted series-parallel model I and model II. Based on the coherent property of MS weighted series-parallel systems, MS weighted series-parallel systems can be decomposed into binary series-parallel systems, and then the binary series-parallel system reliability evaluation approaches can be applied to evaluating the reliability of the corresponding MS weighted series-parallel system.

(2) MS weighted *k*-out-of-*n* systems

Based on different practical situations, two MS weighted *k*-out-of-*n* system models are developed – the so-called MS weighted *k*-out-of-*n* model I and model II. The detailed definitions of these models, proposed UGF algorithms and efficient recursive algorithms to evaluate the system reliability are given, and these two approaches compared with each other.

In **Optimal Design** of MS weighted systems, the proposed research will focus on the three topics below:

(1) Optimal design of binary weighted *k*-out-of-*n* systems

Based on the efficient reliability evaluation algorithms which have already been developed, we can investigate the issue of optimal design of binary weighted *k*-out-of-*n* systems to minimize total system cost or maximize system reliability. Two optimal models are formulated in detail. One is to minimize the expected total cost, and at the same time, guarantee the system reliability greater than a pre-specified value; the other is to maximize system reliability with the constraints on the expected total system cost. Genetic Algorithm (GA) and Tabu Search (TS) methods are both used to solve the resulting optimization models. Since the key to a good TS algorithm is usually quite problem-specific policies and memory structures for deciding what a move is and how long moves are tabu after leading to a local minimum, there is no existing general TS tool available. As a result, many more details are given regarding the TS approach used in this thesis than regarding the GA approach. The results obtained using these two methods are compared. The results show that both are powerful tools for solving these kinds of problems, but TS is more efficient.

(2) Optimal design of MS weighted *k*-out-of-*n* systems

A widely studied reliability optimization problem is the "component selection problem", which involves selection of components with known reliability and cost characteristics for configuring the system. Less adequately addressed has been the problem of determining system cost and utility based on the relationship between component reliability, cost and utility. This so-called "component design problem" has been addressed under this topic. All the optimization problems dealt with under this topic can

be categorized as either minimizing the expected total system cost subject to system reliability requirements, or maximizing system reliability subject to total system cost limitations. The resulting optimization problems are too complicated to be solved by traditional optimization approaches, therefore, Genetic Algorithm (GA) is used to solve them. The results show that GA is a powerful tool for solving these kinds of problems.

(3) Optimal design of MS weighted series-parallel systems

The research on component design problems for multi-state weighted series-parallel systems has been addressed under this topic. Furthermore, comparing to the traditional single-objective optimization model, the optimization model studied under this topic is a multi-objective optimization model which is used to minimize investment cost while simultaneously maximizing system performance utility and system reliability. Genetic algorithm is used to solve the physical programming based optimization models. An example is used to illustrate the increased flexibility and effectiveness of the proposed approach over the single-objective optimization method.

Detailed discussion of the proposed research will be presented in the following chapters.

References:

[1]. M. Agarwal and R. Gupta. Homogeneous redundancy optimization in multi-state series-parallel systems: A heuristic approach. IIE Transactions, 39 (3): 277-289, 2007.

[2]. T. Aven. Reliability evaluation of multi-state systems with multi-state components. IEEE Transactions on Reliability, 34: 473-479, 1985.

[3]. T. Aven. On performance-measures for multi-state monotone systems. Reliability Engineering & System Safety, 41(3):259-266, 1993.

[4]. R.E. Barlow and A.S. Wu. Coherent systems with multi-state components. Mathematics of Operations Research, 3(4): 275-281, 1978.

[5]. R.E. Barlow and K.D. Heidtmann. Computing *k*-out-of-*n* system reliability. IEEE Transactions on Reliability, 33(4): 322-323, 1984.

[6]. R. Billinton and R. Allan. Reliability evaluation of power systems. Plenum Press, New York, 1996.

[7]. R. Billinton and L. Wenyuan. Hybrid approach for reliability evaluation of composite generation and transmission systems using Monte-Carlo simulation and enumeration technique. IEE proceedings-C Generation, transmission, and distribution, 138 (3): 233-241, 1991.

[8]. H.W. Block and T.H. Savits. A decomposition for multi-state monotone systems. Journal of Applied Probability, 19(2): 391-402, 1982.

[9]. R.A. Boedigheimer and K.C. Kapur. Customer-driven reliability models for multi-state coherent systems. IEEE Transactions on Reliability, 43(1): 46-50, 1994.

[10]. R.D. Brunelle and K.C. Kapur. Review and classification of reliability measures for multi-state and continuum models. IIE Transactions, 31(12):1171-1180, 1999.

[11]. D. Butler, A complete importance ranking for components of binary coherent systems with extensions to multistate systems, Naval Research Logistics, 26: 556-578, 1979.

[12]. L. Caldarola, Fault-tree analysis with multistate components, Synthesis and Analysis Methods for Safety and Reliability Studies, G. Apostolakis, S. Garriba, and G. Volta eds., Plenum Press, 1980.

[13]. Y. Chen and Q.Y. Yang. Reliability of two-stage weighted *k*-out-of-*n* systems with components in common. IEEE Transactions on Reliability, 54(3): 431-440, 2005.

[14]. L. Davis. Handbook of genetic algorithms. New York: Van Nostrand Reinhold, 1991.

[15]. A.K. Dhingra. Optimal apportionment of reliability and redundancy in series systems under multiple objectives. IEEE Transactions on Reliability, 41: 576-582, 1992.

[16]. E. El-Neweihi, F. Proschan and J. Sethuraman. Multi-state coherent system. Journal of Applied Probability, 15: 675-688, 1978.

[17]. E. El-Neweihi, F. Proschan. Degradable Systems - A Survey of Multi-state System-Theory. Communications in Statistics-Theory and Methods, 13(4): 405-432, 1984.

[18]. E. El-Neweihi, F. Proschan and J. Sethuraman. Optimal allocation of multi-state components. In: Handbook of Statistics, Vol.7: Quality Control and Reliability. Edited by P.R.Krishnaiah, C.R.Rao. North-Holland, Amsterdam, pp.427-432, 1988.

[19]. M. Finkelstein, Probabilistic approach to some problems of system safety, Microelectron. Reliab. 34, 1994, pp. 1441-1457.

[20]. W. Griffith. Multi-state Reliability Models. Journal of Applied Probability, 17: 735-744, 1980.

[21]. R. Gupta and M. Agarwal. Penalty guided genetic search for redundancy optimization in multi-state series-parallel power system. Journal of Combinational Optimization. 12 (3): 257-277, 2006.

[22]. U. Gurler and A. Kaya. A maintenance policy for a system with multi-state components: an approximate solution. Reliability Engineering & System Safety, 76: 117-127, 2002.

[23]. Y. Higashiyama, 2001, A factored reliability formula for weighted *k*-out-of-*n* system, Asia-Pacific Journal of Operational Research, 18(1): 61-66.

[24]. W.M. Hirsch, M. Meisner and C. Boll. Cannibalization in Multi-component Systems and Theory of Reliability. Naval Research Logistics, 15(3): 331-360, 1968.

[25]. H.Z. Huang. Fuzzy multi-objective optimization decision-making of reliability of series system. Microelectronics and Reliability, 37(3): 447-449, 1997.

[26]. H.Z. Huang, Y.K. Gu, X. Du. An interactive fuzzy multi-objective optimization method for engineering design. Engineering Applications of Artificial Intelligence, 19(5): 451-460, 2006.

[27]. H.Z. Huang, Z. Tian and Y. Gu. Reliability and redundancy apportionment optimization using interactive physical programming. International Journal of Reliability, Quality and Safety Engineering, 11: 213-222, 2004.

[28]. H.Z. Huang, Z. Tian and M.J. Zuo. Intelligent interactive multi-objective optimization method and its application to reliability optimization. IIE Transactions, 37(11): 983-993, 2005.

[29]. J. Huang. Multi-state system reliability analysis. Ph.D. Thesis, University of Alberta, 2001.

[30]. J. Huang and M.J. Zuo. Dominant multi-state systems. IEEE Transactions on Reliability, 53(3): 362-368, 2004.

[31]. J. Huang, M.J. Zuo and Z. Fang. Multi-state consecutive *k*-out-of-*n* systems. IIE Transactions, 35(6): 527-534, 2003.

[32]. J. Huang, M.J. Zuo and Y.H. Wu. Generalized multi-state *k*-out-of-*n*: G systems. IEEE Transactions on Reliability, 49(1): 105-111, 2000.

[33]. J.C. Hudson and K.C. Kapur. Reliability Theory for Multi-state Systems with Multi-state Components. Microelectronics and Reliability, 22(1): 1-7, 1982.

[34]. J.C. Hudson and K.C. Kapur. Reliability analysis for multi-state systems with multi-state components. IIE Transactions, 15, 127-135, 1983.

[35]. J.C. Hudson and K.C. Kapur. Modules in coherent multi-state systems. IEEE Transactions on Reliability, 32: 183-185, 1983.

[36]. J.C. Hudson and K.C. Kapur. Reliability bounds for multi-state systems with multi-state components. Operations Research, 33(1): 153-160, 1985.

[37]. T. Inagaki, K. Inoue and H. Akashi. Interactive optimization of system reliability under multiple objectives. IEEE Transaction on Reliability, 27: 264-267, 1978.

[38]. A. Konak, D.W. Coit and A.E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. . Reliab Engng Syst Safety, 91 (9): 992-1007, 2006.

[39]. W. Kuo and V.R. Prasad. An annotated overview of system-reliability optimization. IEEE Transactions on Reliability, 49(2): 176-87, 2000.

[40]. W. Kuo, V.R. Prasad, F.A. Tillman and C.L. Hwang. Optimal reliability design: fundamentals and applications. Cambridge: Cambridge University Press, 2001.

[41]. W. Kuo and M.J. Zuo. Optimal Reliability Modeling: Principles and Applications. John Wiley & Sons, New York, 2003.

[42]. M. Lanus, L. Yin and K. Trivedi. Hierarchical composition and aggregation of state-based availability and performability models. IEEE Trans. On Reliability. 52: 44-52, 2003.

[43]. Levitin G (Ed.). Computational Intelligence in Reliability Engineering. New Metaheuristics, Neural and Fuzzy Techniques in Reliability, Series: Studies in Computational Intelligence, Vol. 40, Springer-Verlag, 2006.

[44]. G. Levitin. Universal Generating Function in Reliability Analysis and Optimization. Springer-Verlag, 2005.

[45]. G. Levitin. Optimal multilevel protection in series-parallel systems. Reliab Engng Syst Safety, 81 (1): 93-102, 2003.

[46]. G. Levitin. Maximizing survivability of acyclic transmission networks with multi-state retransmitters and vulnerable nodes. Reliab Engng Syst Safety, 77: 189–199, 2002.

[47]. G. Levitin. Redundancy optimization for multi-state system with fixed resource-requirements and unreliable sources. IEEE Transactions on Reliability, 50: 52-59, 2001.

[48]. G. Levitin. Incorporating common-cause failures into series–parallel multi-state system analysis. IEEE Trans Reliab, 50 (4): 380–388, 2001.

[49]. G. Levitin. Multi-state series-parallel system expansion scheduling subject to availability constraints. IEEE Transactions on Reliability, 49: 71-79, 2000.

[50]. G. Levitin and A. Lisnianski. Optimizing survivability of vulnerable series–parallel multi-state systems. Reliab Engng Syst Safety, 79: 317–329, 2002.

[51]. G. Levitin and A. Lisnianski. Optimal separation of elements in vulnerable multi-state systems. Reliab Engng Syst Safety, 73: 55–66, 2001.

[52]. G. Levitin and A. Lisnianski. Structure optimization of multi-state system with two failure modes. Reliability Engineering & System Safety, 72: 75-89, 2001.

[53]. G. Levitin and A. Lisnianski. A new approach to solving problems of multi-state system reliability optimization. Quality and Reliability Engineering International, 17 (2): 93-104, 2001.

[54]. G. Levitin and A. Lisnianski. Survivability maximization for vulnerable multi-state system with bridge topology. Reliab Engng Syst Safety, 70: 125–140, 2000.

[55]. G. Levitin and A. Lisnianski. Optimal multistage modernization of power system subject to reliability and capacity requirements. Electrical Power Systems Research, 50: 183-190, 1999.

[56]. G. Levitin and A. Lisnianski. Joint redundancy and maintenance optimization for multi-state series-parallel systems. Reliability Engineering & System Safety, 64(1): 33-42, 1998.

[57]. G. Levitin, A. Lisnianski, H. Ben Haim and D. Elmakis. Redundancy optimization for series-parallel multi-state systems. IEEE Transactions on Reliability, 47(2): 165-172, 1998.

[58]. G. Levitin, A. Lisnianski and D. Elmakis. Structure optimization of power system with different redundant elements. Electric Power Systems Research, 43 (1): 19-27, 1997.

[59]. J.A. Li, Y. Wu, K.K. Lai and K. Liu. Reliability estimation and prediction of multi-state components and coherent systems. Reliability Engineering and System Safety, 88 (1): 93-98, 2005

[60]. W. Li and M.J. Zuo. Reliability evaluation of multi-state weighted *k*-out-of-*n* systems. Reliability Engineering and System Safety, (available online) 93(1): 161-168, 2008.

[61]. J.S. Lin, C.C. Jan and J. Yuan. On reliability evaluation of a capacitated flow network in terms of minimal pathsets. Networks, 25: 131-138, 1995.

[62]. A. Lisnianski and G. Levitin. Multi-state System Reliability: Assessment, Optimization and Applications. World Scientific, Singapore, 2003.

[63]. A. Lisnianski, G. Levitin, H. Ben-Haim and D. Elmakis. Power system structure optimization subject to reliability constraints. Electric Power Systems Research, 39 (2): 145-152, 1996.

[64]. P.X. Liu, M.J. Zuo and M. Q-H Meng. A neural network approach to optimal design of continuous-state parallel-series systems. Computers and Operations Research, 30: 339-352, 2003.

[65]. Y. Massim, R. Meziane, A. Zeblah and M. Rahli. Ant colony optimization for multi-state series-parallel system expansion scheduling. Electrical Engineering. 87 (6): 327-336, 2005.

[66]. Y. Massim, A. Zeblah, R. Meziane, M. Benguediab and A. Ghouraf. Optimal design and reliability evaluation of multi-state series-parallel power systems. Nonlinear Dynamics. 40 (4): 309-321, 2005.

[67]. F. Meng. More on optimal allocation of components in coherent systems. Journal of Applied Probability, 33: 548-556, 1996.

[68]. M.C. Meng. Comparing two reliability upper bounds for multi-state systems. Reliability Engineering and System Safety, 87(1): 31-36, 2005.

[69]. A. Messac. Physical Programming: effective Optimization for Computational Design. AIAA Journal, 34(1): 149-158, 1996.

[70]. A. Messac and B. Wilson. Physical Programming for Computational Control. AIAA Journal, 36: 219-226, 1998.

[71]. A. Messac, M. Martinez and T. Simpson. Introduction of a Product Family Penalty Function using Physical Programming. ASME Journal of Mechanical Design, 124: 164-172, 2002.

[72]. A. Messac and A. Ismail-Yahaya. Multiobjective Robust Design using Physical Programming. Structural and Multidisciplinary Optimization, Journal of the International Society of Structural and Multidisciplinary Optimization (ISSMO), 23: 357-371, 2002.

[73]. R. Meziane, Y. Massim, A. Zeblah, A. Ghoraf and R. Rahli. Reliability optimization using ant colony algorithm under performance and cost constraints. Electric Power Systems Research. 76 (1-3): 1-8, 2005.

[74]. K.B. Misra and M.D. Ljubojevic. Optimal reliability design of a system: a new look. IEEE Transactions on Reliability, 22(5): 255-258, 1973.

[75]. B. Natvig. Two suggestions of how to define a multi-state coherent system. Applied Probability, 14: 391-402, 1982.

[76]. B. Natvig and A. Streller, The steady-state behavior of multistate monotone systems, J. Applied Probability, vol. 21, 1984, pp 826-835.

[77]. E. Nikolaidis, D.M. Ghiocel and S. Singhal. Engineering Design Reliability Handbook, CRC Press, 2004.

[78]. M. Nourelfath and D. Ait-Kadi. Optimization of series-parallel multi-state systems under maintenance policies. Reliability Engineering and System Safety, 92 (12): 1620-1626, 2007.

[79]. M. Nourelfath and Y. Dutuit. A combined approach to solve the redundancy optimization problem for multi-state systems under repair policies. Reliability Engineering and System Safety, 86(3): 205–213, 2004.

[80]. P.D.T. O'Connor. Practical Reliability Engineering, John Wiley & Sons, 2002.

[81]. K.S. Park. Fuzzy apportionment of system reliability. IEEE Transaction on Reliability, 36: 129-132, 1987.

[82]. M. Patel, K.E. Lewis, A. Maria and A. Messac. System Design through Subsystem Selection using Physical Programming. AIAA Journal, 41: 1089-1096, 2003.

[83]. H. Pham, Reliability Modeling, Analysis and Optimization, World Scientific, 2006.

[84]. L. Podofillini1, E. Zio and M. Marella. A multi-state Monte Carlo simulation model of a railway network system. Advances in Safety and Reliability, Kołowrocki (ed.), Taylor & Francis Group, London. pp. 1567-1575, 2005.

[85]. S.S. Rao and A.K. Dhingra. Reliability and redundancy apportionment using crisp and fuzzy multiobjective optimization approaches. Reliability Engineering and System Safety, 37: 253-261, 1992.

[86]. V. Ravi, P.J. Reddy and H.J. Zimmermann. Fuzzy global optimization of complex system reliability. IEEE Transactions on Fuzzy System, 8: 241-248, 2000.

[87]. J.E. Ramirez-Marquez and D.W. Coit. A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems. Reliability Engineering & System Safety, 83 (3): 341-349, 2004.

[88]. J.E. Ramirez-Marquez and D.W. Coit. A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability. Reliability Engineering & System Safety, 87: 253-264, 2005.

[89]. J.E. Ramirez-Marquez, D.W. Coit and M. Tortorella. A generalized multi-state based path vector approach for multi-state two-terminal reliability. IIE Transactions, 38 (6): 477-488, 2006.

[90]. S. Ross. Multi-valued State Component Systems. Annals of Probability, 7: 379-383, 1979.

[91]. A.M. Rushdi. Utilization of symmetric switching functions in the computation of *k*-out-of-*n* system reliability. Microelectronics and Reliability, 26: 973-987, 1986.

[92]. M. Sakawa. Multi-objective optimization by the surrogate worth tradeoff method. IEEE Transaction on Reliability, 27: 311-314, 1978.

[93]. S. Satitsatian and K.C. Kapur. An algorithm for lower reliability bounds of multi-state two-terminal networks. IEEE Transactions on Reliability, 55(2): 199-206, 2006.

[94]. J.H. Saleh and K. Marais. Highlights from the early (and pre-) history of reliability engineering. Reliability Engineering & System Safety, 91(2): 249-256, 2006.

[95]. H. Taboada, F. Baheranwala, D. Coit and N. Wattanapongsakorn. Practical solutions for multi-objective optimization: an application to system eeliability design problems. Reliability Engineering & System Safety, 92(3): 314-322, 2007.

[96]. Z. Tian and M.J. Zuo. Redundancy allocation for multi-state systems using physical programming and genetic algorithms. Reliability Engineering & System Safety, 91(9): 1049-1056, 2006.

[97]. Z. Tian and M.J. Zuo. A unified model of *k*-out-of-*n* systems and its reliability evaluation. Proceedings of the 2007 Industrial Engineering Research Conference, Nashville, Tennessee, USA, May 19-23, 2007. pp. 1770-1775.

[98]. Z. Tian, R. CM Yam, M.J. Zuo and H. Huang. Reliability bounds for multi-state *k*-out-of-*n* systems. IEEE Transactions on Reliability, 2006. Accepted for Publication.

[99]. Z. Tian, M.J. Zuo and H. Huang. Reliability-redundancy allocation for multi-state series-parallel systems. IEEE Transactions on Reliability, Accepted for Publication, 2007 .

[100]. Z. Tian, M.J. Zuo and H. Huang. Reliability-Redundancy Allocation for Multi-State Series-Parallel Systems. Proceedings of the 2005 European Safety & Reliability Conference, Tri City, Poland, June 20-23, 2005. pp. 1925-1930.

[101]. Z. Tian, M.J. Zuo and R. CM Yam. Performance Evaluation for Generalized Multi-State *k*-out-of-*n* Systems. Proceedings of the 2005 Industrial Engineering Research Conference, Atlanta, USA, May 14-18, 2005.

[102]. Z. Tian, M.J. Zuo, R. C.M. Yam. The Multi-State *k*-out-of-*n* Systems and Their Performance Evaluation. IIE Transactions, Submitted, 2006..

[103]. I. Ushakov. Universal generating function. Soviet Journal Computer Systems Science, 24(5): 118–129, 1986.

[104]. I. Ushakov. Optimal standby problems and a universal generating function. Sov J Comput Syst Sci, 25 (4): 79–82, 1987

[105]. J. K. Vaurio Reliability and availability equations for multi-state components. Reliability Engineering, 7: 1-19, 1984.

[106]. M. Veeraraghavan and K.S. Trivedi, A Combinatorial Algorithm for Performance and Reliability Analysis Using Multistate Models, IEEE Transactions on Computers, 43: 226-229, 1994.

[107]. A. Wood, Multistate Block Diagrams and Fault Trees. IEEE Transactions on Reliability. R-34 (3): 236-240, 1985.

[108]. S.M. Wu and L.Y. Chan. Performance utility-analysis of multi-state systems. IEEE Transactions on Reliability. 52 (1): 14-21, 2003

[109]. J.S. Wu and R.J. Chen. An algorithm for computing the reliability of a weighted-*k*-out-of-*n* system. IEEE Transactions on Reliability, 43: 327-328, 1994.

[110]. J. Xue and K. Yang, Dynamic Reliability Analysis of Coherent Multistate Systems, IEEE Transactions on Reliability, 44 (4): 683-688, 1995.

[111]. H. Yamamoto, M.J. Zuo, T. Akiba and Z.G. Tian. Recursive formulas for the reliability of multi-state consecutive *k*-out-of-*n*: G Systems. IEEE Transactions on Reliability, 55 (1): 98-104, 2006.

[112]. W.C. Yeh. Multi-state node acyclic network reliability evaluation. Reliability Engineering & System Safety, 78 (2): 123-129, 2002.

[113]. W.C. Yeh. An evaluation of the multi-state node networks reliability using the traditional binary-state networks reliability algorithm. Reliability Engineering & System Safety, 81 (1): 1-7, 2003.

[114]. X.Y. Zang, H.R. Sun and K.S. Trivedi. A BDD-based algorithm for reliability analysis of phased-mission systems. IEEE Transactions on Reliability, 48(1): 50-60, 1999.

[115]. E. Zio, M. Marella and L. Podofillini. A Monte Carlo simulation approach to the availability assessment of multi-state systems with operational dependencies. Reliability Engineering & System Safety, 92 (7): 871-882, 2007.

[116]. E. Zio and L. Podofillini. Monte Carlo simulation analysis of the effects of different system performance levels on the importance of multi-state components. Reliability Engineering & System Safety, 82: 63-73, 2003.

[117]. E. Zio, L. Podofillini and G. Levitin. Estimation of the importance measures of multi-state elements by Monte Carlo simulation. Reliability Engineering & System Safety. 86 (3): 191-204, 2004.

[118]. M.J. Zuo and M. Liang. Reliability of multi-state consecutively-connected systems. Reliability Engineering & System Safety, 44: 173-176, 1994.

[119]. M.J. Zuo, B. Liu and D. Murthy. Replacement-repair policy for multi-state deteriorating products under warranty. European Journal of Operational Research, 123: 519-530, 2000.

[120]. M.J. Zuo and Z. Tian. Performance evaluation for generalized multi-state *k*-out-of-*n* systems. IEEE Transactions on Reliability, 55(2): 319-327, 2006.

[121]. M.J. Zuo and Z. Tian. An efficient method for reliability evaluation of multi-state networks given all minimal path vectors. IIE Transactions, 39(8): 811-817, 2007.

Chapter 3

Fundamentals of Multi-State Weighted System Reliability

This chapter presents fundamental knowledge of multi-state weighted reliability, including some basic concepts and tools such as structure function, state distribution, and utility distribution. Typical system structures are discussed, including series-parallel systems and k-out-of-n systems. The general framework of recursive algorithms and universal generating functions are also presented in this chapter.

3.1 Basic concepts

- **Structure function.** A multi-state component or system can be in M+1 possible states: {0, 1, 2, …, M}. There is a utility value corresponding to each state. Suppose a system has n components. We use a vector, $u = (u_1, u_2, \ldots, u_n)$, to represent the component utilities, where component i has utility u_i. "Structure function" represents the relationship between the component utilities and the system utility. $\gamma(u)$ denotes the system utility as a function of the component utilities.

- **State distribution.** In binary reliability theory, the reliability of a component or a system actually refers to its probability of being in state 1. In the context of multi-state reliability, there are more than two possible states. We use "state distribution" instead of "reliability" to denote the probability of a multi-state component being at different state levels.

- **Utility distribution.** In a binary weighted system, component i carries a utility of u_i, $u_i > 0$. In a MS weighted system, a component has more than two states. There is a utility corresponding to every state of the component. So, corresponding to the state distribution, the "utility distribution" is used to denote the performance of a multi-state component at different performance levels. One point we need to mention is that the "utility" used here has the same meaning as the "weight" Wu and Chen used in their paper [12]. They all mean the contribution of the component. In the binary weighted context, the component that makes a higher contribution to a system has a higher "weight". The component that contributes less has a lower "weight". Since we also discuss the "physical weight" of the component, in order to distinguish the "weight" from "physical weight", clearly we use "utility" as a substitute for the word "weight". The notation, w_i, which was used by Wu and Chen [12] for the component "weight" is used as the "physical weight" of the component in this research. Consistent with the binary weighted system, the MS weighted system uses the "utility distribution" to denote the contribution of a multi-state component at different performance levels.

3.2 Typical system structures

In order to conduct the reliability evaluation and optimal design, we first need to look into the logical relationships among components, that is to say, to identify the system structure. In this section, some typical system structures are presented. For both the series-parallel system and the k-out-of-n system, we start from the traditional binary model, and then move on to the MS system model which has already been studied, finally bringing forward the MS weighted model which we will study in this thesis. We also

introduce a special case of MS weighted *k*-out-of-*n* system – the binary weighted *k*-out-of-*n* system. The reliability evaluation for it has already been reported by Wu and Chen [12]. In this thesis, the reliability optimal design of it will be studied. For each system structure, first we discuss it with regard to the binary reliability framework where the system and the components can take only two possible states — state 1 (working state) and state 0 (failed state) — and then with regard to the multi-state reliability framework, and finally the multi-state weighted reliability framework. Finally the system structure is discussed with regard to the multi-state weighted reliability framework. We focus our discussion on the two typical structures of multi-state systems to be studied in this thesis work: series-parallel systems and *k*-out-of-*n* systems.

3.2.1 Series-parallel systems

(1) Binary series-parallel systems: A binary series-parallel system has N subsystems connected in series, and each subsystem is a parallel system [7]. A parallel system is working (in state 1) as long as at least one of its components is working (in state 1). In other words, a parallel system is failed (in state 0) if all of its components are failed (in state 0). A series system is working (in state 1) if all of its components are working (in state 1).

(2) Multi-state series-parallel systems: The multi-state series-parallel system defined by Barlow and Wu [1] has been widely studied. A multi-state series-parallel system consists of subsystems, S_1 to S_N, connected in series. Each subsystem, say S_i, has some components connected in parallel. The following assumptions are used: (1) The components in a subsystem are independent. (2) The components and the system may be in M+1 possible states, namely, 0, 1, 2, ..., M. (3) The multi-state series-parallel

systems under consideration are coherent systems. According to the multi-state system definition of Barlow and Wu [1], the state of a parallel system is defined as being the state of the best component in the system, and the state of a series system is defined as being the state of the worst component in the system.

(3) Multi-state weighted series-parallel systems:

When a MS series-parallel system is extended to the MS weighted series-parallel system, the system is still a coherent system and consists of subsystems, S_1, S_2, ..., S_N, connected in series. Each subsystem, say S_i, has n_i different MS weighted components connected in parallel. In a subsystem, each component may be in $M+1$ states: $\{0,1,2,\cdots,M\}$. Component i has a probability, p_{ij}, of being in state j, and there is a utility value, u_{ij}, corresponding to each component state. The utility value, u_{ij}, describes the component performance capability in that state.

The structure of a multi-state weighted series-parallel system is as shown in Figure 3.1.

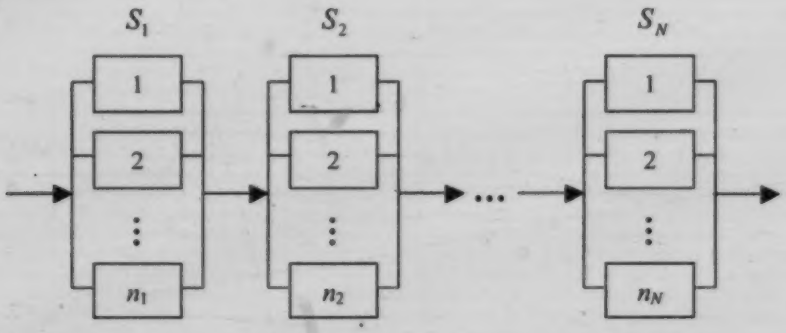


Figure 3.1: Structure of a multi-state weighted series-parallel system

3.2.2 K-out-of-n systems

(1) **Binary *k*-out-of-*n* systems:** The *k*-out-of-*n* system structure is a very popular type of redundancy in fault tolerant systems, with wide applications in both industrial and military systems. An *n*-component system is called a binary *k*-out-of-*n*:G system if it is working whenever at least *k* components are working. An *n*-component system is called a binary *k*-out-of-*n*:F system if it is failed whenever at least *k* components are failed. Efficient reliability evaluation algorithms for binary *k*-out-of-*n* systems with independent components have been provided by Barlow and Heidtmann [1] and Rushdi [9].

(2) **Multi-state *k*-out-of-*n* systems:** Binary *k*-out-of-*n* system models have been extended to multi-state *k*-out-of-*n* system models by allowing components and systems to take more than two possible states [4] and [5]. From Huang et al. [6], an *n*-component system is called a generalized multi-state *k*-out-of-*n*:G system if $\phi(x) \geq j,\ (1 \leq j \leq M)$ whenever there exists an integer value, $l(j \leq l \leq M)$, such that at least k_l components are in state *l* or above.

(3) **Multi-state weighted *k*-out-of-*n* systems:** A special case of multi-state weighted *k*-out-of-*n* systems is the binary weighted *k*-out-of-*n* system which has already been developed by Wu and Chen [12]. In a binary weighted *k*-out-of-*n*:G system, component *i* carries a positive integer weight, w_i, $w_i > 0$, for $i = 1, 2, \ldots, n$. The total weight of all components is w, $w = \sum_{i=1}^{n} w_i$. The system works if and only if the total weight of working components is at least *k*, a pre-specified value. Since *k* is a weight, it may be greater than *n* because *k* and *n* are expressed using different units of measurement. Such a binary weighted *k*-out-of-*n*:G system is equivalent to a binary

weighted $(w-k+1)$-out-of-n: F system wherein the system fails if and only if the total weight of failed components is at least $w-k+1$. Using these generalizations, the binary k-out-of-n system model is a special case of the binary weighted k-out-of-n system model because each component in a k-out-of-n system has a weight of one. As mentioned before, the "weight" used in Wu and Chen's paper [12] is the same meaning as the "utility" used in this thesis. A multi-state weighted k-out-of-n system is a system with n multi-state weighted components. Each component has multiple possible states with multiple levels of capacity. Component i has a probability, p_{ij}, of being in state j, and there is a utility value, u_{ij}, corresponding to each component state. The capacity of the system described by the system utility is equal to the sum of the utilities of all the components or of some special kinds of components in the system. The detailed definitions of two types of multi-state weighted k-out-of-n systems will be presented in Chapter 4.

3.3 Recursive Algorithms

Recursive algorithms have been efficient means of making system reliability evaluations [3]. There are three fundamental elements in a recursive algorithm. Below, the recursive algorithm for binary weighted k-out-of-n systems [12] will be used to illustrate these elements.

Below, $R(k, n)$ represents the probability that a system with n components can provide a total utility of at least k. Then, $R(k, n)$ is the reliability of the weighted k-out-of-n system. The following recursive equation can be used for evaluating the reliability of such systems.

$$R(k,n) = p_n R(k-u_n, n-1) + q_n R(k, n-1), \tag{3.1}$$

which requires the following boundary conditions:

$$R(k,n) = 1, \text{ for } k \le 0,\ n \ge 0, \tag{3.2}$$

$$R(k,0) = 0, \text{ for } k > 0. \tag{3.3}$$

As described in Tian's thesis [10], a recursive algorithm has the following three key elements:

(1) **Recursive function** is the key function that calls itself in the recursive algorithm. It has one or multiple input parameters, which change during the course of the recursive algorithm. A recursive function with a set of input parameters is calculated via several recursive functions with sets of simpler input parameters. In the recursive algorithm for binary weighted *k*-out-of-*n* systems in (3.1) to (3.3), $R(n, k)$ is the recursive function, in which n and k are the parameters. As shown in the equations, the recursive function, $R(n, k)$, is calculated via the recursive functions $R(k\text{-}u_n, n\text{-}1)$ and $R(k, n\text{-}1)$.

(2) **Updating algorithms.** The updating algorithm decides how the recursive function calls itself. In other words, the updating algorithm decides the relationship between a recursive function with certain parameters and recursive functions with different parameters, which are typically less complex. In the recursive algorithm for binary weighted *k*-out-of-*n* systems, Equation (3.1) shows the updating algorithm, that is, how to calculate the recursive function, $R(n, k)$, via two recursive functions, $R(k\text{-}u_n, n\text{-}1)$ and $R(k, n\text{-}1)$, with smaller input parameter values.

(3) Boundary conditions. When one of the boundary conditions is met, the recursive function will take a certain value, or can be determined in a specific and simple way. In the recursive algorithm for binary weighted k-out-of-n systems, Equations (3.2) and (3.3) show the boundary conditions. There are two boundary conditions in this case. The first one is when $k \leq 0,\ n \geq 0,$, the value of the recursive function $R(n, k)$, is 1. The other boundary condition is when $k>0$, the value of the recursive function $R(k, 0)$, is 0.

3.4 Universal Generating Functions

The universal generating function (UGF) was introduced by Ushakov [11] in 1986. In 1996, Lisnianski et al. [8] applied the UGF to power system reliability analysis. The UGF technique allows one to find the entire system performance distribution based on the performance distributions of its components by using a rapid algebraic procedure. Although the UGF is a universal technique, specific operators for different kinds of systems need to be developed based on their specific logical structures. Since the binary weighted k-out-of-n system is a special case of the multi-state weighted k-out-of-n system, below, the binary weighted k-out-of-n system is used to illustrate how to use the UGF to evaluate system reliability.

In a binary weighted k-out-of-n system, the UGF for the components is:

$$U_i(z) = (1 - p_i)z^0 + p_i z^{u_i}. \qquad (3.4)$$

where p_i is the reliability of component i, u_i is the utility of component i in the binary weighted k-out-of-n system, and z is the transform parameter used to express the UGF in

a form of moment–generating functions.

To obtain the UGF of a general system based on the UGFs of theist individual components, the following composition operator, Ω, can be used [8]:

$$U_s(z) = \Omega(U_1(z), U_2(z) \cdots, U_n(z)), \tag{3.5}$$

where Ω satisfies the following general conditions:

$$\Omega(U_1(z) \cdots U_k(z), U_{k+1}(z) \cdots U_n(z)) = \Omega(U_1(z) \cdots U_{k+1}(z), U_k(z) \cdots U_n(z)) \tag{3.6}$$

$$\Omega(U_1(z) \cdots U_k(z), U_{k+1}(z) \cdots U_n(z)) = \Omega(\Omega(U_1(z) \cdots U_k(z)), \Omega(U_{k+1}(z) \cdots U_n(z))) \tag{3.7}$$

Where there are only two elements $U_1(z)$ and $U_2(z)$, we have:

$$\Omega(U_1(z), U_2(z)) = \Omega[\sum_{j=1}^{J} p_{1j} z^{g_{1j}} \cdot \sum_{l=1}^{L} p_{2l} z^{g_{2l}}] = \sum_{j=1}^{J} \sum_{l=1}^{L} p_{1j} p_{2l} z^{(g_{1j} + g_{2l})}, \tag{3.8}$$

where J and L are the numbers of possible performance levels for element $U_1(z)$ and $U_2(z)$. Note that $U_1(z)$ and $U_2(z)$ may be the UGFs of two individual components, and they may also be the UGFs of two subsystems.

Having a binary weighted k-out-of-n output performance distribution in the above form, one can obtain the system reliability for the arbitrary k using the following operator, δ_A:

$$R_s(k) = \delta_A(U_s(z),\ k) = \delta_A(\sum_{i=1}^{h} p_i z^{G_i},\ k) = \sum_{i=1}^{h} p_i \alpha(G_i - k). \tag{3.9}$$

where h is the number of possible performance levels in the polynomial $U_s(z)$.

The function $\alpha(x)$ in the above equation means:

$$\alpha(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.10}$$

In this chapter, some basic concepts and tools such as structure function, state distribution, and utility distribution are presented. Typical system structures, including series-parallel systems and *k*-out-of-*n* systems, from the traditional binary system to the MS weighted system are all introduced. The general framework of recursive algorithms and universal generating functions (UGF) are also described in this chapter. In the following chapters, these concepts, structures and approaches will be used to define the MS weighted system model and evaluate the system reliability. The recursive and UGF methods will both be used to evaluate the system reliability distribution and will be compared with each other on the computation efficiency.

References:

[1]. R.E. Barlow and K.D. Heidtmann. Computing *k*-out-of-*n* system reliability. IEEE Transactions on Reliability, 33(4): 322-323, 1984.

[2]. R.E. Barlow and A.S. Wu. Coherent systems with multi-state components. Mathematics of Operations Research, 3(4): 275-281, 1978.

[3]. R.D. Brunelle and K.C. Kapur. Review and classification of reliability measures for multi-state and continuum models. IIE Transactions, 31(12):1171-1180, 1999.

[4]. E. El-Neweihi, F. Proschan and J. Sethuraman. Multi-state coherent system. Journal of Applied Probability, 15: 675-688, 1978.

[5]. J. Huang. Multi-state system reliability analysis, Ph.D. Thesis, University of Alberta, 2001.

[6]. J. Huang, M.J. Zuo and Y.H. Wu. Generalized multi-state *k*-out-of-*n*: G systems. IEEE Transactions on Reliability, 49(1): 105-111, 2000.

[7]. W. Kuo and M.J. Zuo. Optimal Reliability Modeling: Principles and Applications. John Wiley & Sons, New York, 2003.

[8]. A. Lisnianski, G. Levitin, H. Ben-Haim and D. Elmakis. Power system structure optimization subject to reliability constraints. Electric Power Systems Research, 39 (2): 145-152, 1996.

[9]. A.M. Rushdi. Utilization of symmetric switching functions in the computation of *k*-out-of-*n* system reliability. Microelectronics and Reliability, 26: 973-987, 1986.

[10]. Z. Tian. Multi-State System Reliability Analysis and Optimization. PhD Thesis, University of Alberta, 2007.

[11]. I. Ushakov. Universal generating function. Soviet Journal Computer Systems Science, 24(5): 118–129, 1986.

[12]. J.S. Wu and R.J. Chen. An algorithm for computing the reliability of a weighted-*k*-out-of-*n* system. IEEE Transactions on Reliability, R-43: 327-328, 1994.

Chapter 4

Reliability Evaluation of Multi-state Weighted *k*-out-of-*n* Systems

The *k*-out-of-*n* systems have been extensively studied in recent years [14]. A binary weighted *k*-out-of-*n* model, which is a special case of multi-state weighted *k*-out-of-*n* model, has also been reported in the literature [18]. In this chapter, a more general system – MS weighted *k*-out-of-*n* system is brought forward. Two kinds of reliability evaluation methods: recursive method and universal generating function (UGF) method are used to evaluate the binary and MS weighted *k*-out-of-*n* system reliability. Since the modern systems become more and more complex, efficient reliability evaluation method is highly needed. The reason we used two methods to evaluate the system reliability is to compare them and find the more efficient one. We first compare these two approaches for reliability evaluation of binary weighted *k*-out-of-*n* systems. We then provide two models of multi-state weighted *k*-out-of-*n* system. Recursive algorithms are presented for reliability evaluation of these new models and then compared with the universal generating function (UGF) method. A shorter version of materials in this chapter has been published in [11].

4.1 Binary weighted k-out-of-n: G systems

In a binary *k*-out-of-*n*:G system, the system works if and only if at least k components work. In a binary *k*-out-of-*n*:F system, the system fails if and only if at least k components fail. Wu and Chen [18] generalized the binary *k*-out-of-*n* system models into

the binary weighted k-out-of-n models. In a binary weighted k-out-of-n:G system, component i carries a utility of w_i, $w_i > 0$ for $i = 1, 2, \cdots, n$. Here, the utility of each component is actually representing the utility of the component. The total utility of all components is w, $w = \sum_{i=1}^{n} w_i$. The system works if and only if the total utility of working components is at least k, a pre-specified value. Since k is a utility, it may be larger than n because they have different measuring units. Such a binary weighted k-out-of-n:G system is equivalent to a binary weighted $(w-k+1)$-out-of-n:F system wherein the system fails if and only if the total utility of failed components is at least $w-k+1$. With these generalizations, the binary k-out-of-n system models are special cases of the binary weighted k-out-of-n system models since each component in a binary k-out-of-n system has a utility of one.

To evaluate the reliability of a binary weighted k-out-of-n system, Wu and Chen [18] provided a recursive algorithm. Higashiyama [3] provided a method to express the system reliability of a binary weighted k-out-of-n system in fewer terms than that with the algorithm by Wu and Chen [18]. However, the time complexity and space complexity of these two methods reported in [3] and [18] are the same. The most recently reported study on the binary weighted k-out-of-n systems is by Chen and Yang [1]. In [1], the one-stage binary weighted k-out-of-n model was extended to the two-stage binary weighted k-out-of-n model with components in common among the stages.

The concept of universal generating function (UGF) was introduced by Ushakov [16] in 1986. Detailed mathematical foundations of the UGF method were presented in Gnedenko and Uskakov [2], Ushakov [15] and [17]. In 1996, Lisnianski et al. [11]

applied this method to power system reliability analysis. In a series of research work by Levitin and Lisnianski [9], [9], [5], [12], [6], [8] and [5], the UGF method were used for performance evaluation of several multi-state system structures including series, parallel, series-parallel, and the bridge.

In this section, we will focus on binary systems wherein each component and the system may only be in two possible states: working or failed. The UGF of component i is a polynomial function denoted by $U_i(z)$ that relates the probability of each state to the performance or utility of the component when in that state. For the binary weighted k-out-of-n: G system, the UGF of component i is

$$U_i(z) = p_i z^{w_i} + (1 - p_i)z^0 = p_i z^{w_i} + (1 - p_i), \tag{4.1}$$

where p_i is the reliability of component i and w_i is the utility of component i when it is working.

The UGF of a system is a polynomial function denoted by $U_s(z)$ that defines the system output performance distribution (OPD). It relates the probability P_j of state j to the performance G_j of the system when in state j in the following form:

$$U_s(z) = \sum_{j=1}^{J} P_j z^{G_j}, \tag{4.2}$$

where J is the largest possible state of the system. System state j corresponds to a certain combination of the components' states. The system utility is the sum of utilities of components in those states. To obtain the UGF of a system based on the individual UGFs of the components, we need to use the following operator Ω [15]:

$$U_s(z) = \Omega(U_1(z), U_2(z) \cdots, U_n(z)), \tag{4.3}$$

where n is the number of components in the system.

The Ω operator has the following properties [15]:

$$\Omega(U_1(z), \cdots, U_i(z), U_{i+1}(z), \cdots, U_n(z)) = \Omega(U_1(z), \cdots, U_{i+1}(z), U_i(z), \cdots, U_n(z)),$$
$$\Omega(U_1(z), \cdots, U_i(z), U_{i+1}(z), \cdots, U_n(z)) = \Omega(\Omega(U_1(z), \cdots, U_i(z)), \Omega(U_{i+1}(z), \cdots, U_n(z))),$$
$$\Omega(U_1(z), U_2(z)) = \Omega\left[\sum_{j=1}^{J} p_{1_j} z^{g_{1_j}}, \sum_{l=1}^{L} p_{2_l} z^{g_{2_l}}\right] = \sum_{j=1}^{J} \sum_{l=1}^{L} p_{1_j} p_{2_l} z^{(g_{1_j} + g_{2_l})}.$$

Once the UGF of a binary weighted k-out-of-n: G system is obtained, we can use the following operator δ_A [9] to obtain the reliability of the system for any given arbitrary k value:

$$R_s(k) = \delta_A(U_s(z), k) = \delta_A\left(\sum_{j=1}^{J} P_j z^{G_j}, k\right) = \sum_{j=1}^{J} P_j \alpha(G_j - k), \tag{4.4}$$

where the function $\alpha(x)$ in the above equation is defined as:

$$\alpha(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

Example 4.1: Consider a binary weighted 5-out-of-3: G system. It has three components with utilities 2, 6 and 4, respectively. The system works if and only if the total utility of working components is at least 5. The UGFs for the three components are as follows:

$$U_1(z) = q_1 z^0 + p_1 z^2,$$
$$U_2(z) = q_2 z^0 + p_2 z^6,$$
$$U_3(z) = q_3 z^0 + p_3 z^4.$$

where $q_i = 1 - p_i$, $i = 1,2,3$.

Based on the individual UGFs of the components given above, we can obtain the system UGF by using operator Ω as follows:

$$\begin{aligned} U_s(z) &= \Omega(U_1(z), U_2(z), U_3(z)) \\ &= \Omega((q_1 z^0 + p_1 z^2), (q_2 z^0 + p_2 z^6), (q_3 z^0 + p_3 z^4)) \\ &= q_1 q_2 q_3 z^0 + p_1 q_2 q_3 z^2 + q_1 q_2 p_3 z^4 + (q_1 p_2 q_3 + p_1 q_2 p_3) z^6 \\ &\quad + p_1 p_2 q_3 z^8 + q_1 p_2 p_3 z^{10} + p_1 p_2 p_3 z^{12} \end{aligned}$$

Since $k = 5$, the reliability of the system based on the above expression can be found to be:

$$R_s(5) = \delta_A(U_s(z), 5) = q_1 p_2 q_3 + p_1 q_2 p_3 + p_1 p_2 q_3 + q_1 p_2 p_3 + p_1 p_2 p_3 = p_2 + q_2 p_1 p_3$$

In the following, we provide a comparison of the recursive algorithm by Wu and Chen [18] and the UGF approach for reliability evaluation of the binary weighted *k*-out-of-*n* systems. The computer programs for both approaches were developed in Matlab 7.0. To run these programs we used a Pentium 4 computer with a 3.00GHZ CPU and 512MB RAM under Windows XP operating system. The input data to the programs were k, n, a vector of component weights, and a vector of component reliabilities. The program for Wu and Chen's recursive method included the complete recursive process as described in [18]. The program for the UGF method included the process from building the UGFs for the components to obtaining the system reliability based on the system UGF as illustrated in Example 4.3. Random numbers between 0 and 1 were generated to represent component reliability values and random numbers between 0 and 10 were used to

represent component weights. The CPU times taken by the two approaches for different system sizes are given in Table 4.1.

Table 4.1: CPU time comparison of the Wu and Chen algorithm and the UGF approach as a function of system size n (k=200)

System Size n	CPU Time in Seconds of the Wu and Chen Method	CPU Time in Seconds of the UGF Method
3	0.0100	0.0100
4	0.0100	0.0100
5	0.0100	0.0100
6	0.0100	0.0100
7	0.0150	0.0100
8	0.0160	0.0100
9	0.0310	0.0160
10	0.0630	0.0310
11	0.1090	0.0310
12	0.2030	0.0940
13	0.4060	0.2670
14	0.8130	1.6450
15	1.6400	11.5600
16	3.2970	108.2380
18	13.2340	2005.1821
20	51.4140	More than one hour

From Table 4.1, we find that when the size of the system was very small ($n \leq 6$), the CPU times required by the two approaches were pretty much the same. When the size of the system grew a little bigger, between $n = 7$ and $n = 13$, the UGF method was a little bit faster than the Wu and Chen method. However, when the size of the system grew even bigger, the computation time of the UGF method increased dramatically and became much longer than the CPU time of the Wu and Chen method. When $n = 15$, the CPU time of the UGF method was more than 7 times of the CPU time of the Wu and Chen method. When $n = 18$, the CPU time of the UGF method was more than 100 times of the CPU time of the Wu and Chen method. It is clear that the CPU time advantage of the UGF approach was very minimal even when the system size was moderate. Thus,

generally speaking, the recursive method of Wu and Chen [18] is more efficient than the UGF approach.

4.2 Multi-state weighted k-out-of-n:G system: model I

In Section 4.1, we have described the binary weighted k-out-of-n:G system model. In a binary weighted k-out-of-n:G system, the utility of the system is the sum of the utilities of working components. When a component is failed, its contribution to system utility is 0.

In a multi-state context, a component may be in different states. When it is in a different state, it may have a different contribution to the system. When it is completely failed, its contribution to the system is zero. In the multi-state weighted k-out-of-n:G model to be defined in this section, every component in every possible state has a certain contribution to the system's performance. The formal definition of Model I of the multi-state weighted k-out-of-n:G system is given below.

Definition 4.1: In a system with n components, each component and system may be in $M+1$ possible states: 0, 1, 2, ..., M. Component i ($1 \le i \le n$), when in state j ($0 \le j \le M$), has a utility value of $w_{i,j}$. The system is in state j or above if the total utility of all components is greater than or equal to k_j, a pre-specified value. Let ϕ be the structure function of the system representing the state of the system and W the total utility of all components. Then, this definition means $\Pr\{\phi \ge j\} = \Pr\{W \ge k_j\}$. Since state 0 is the worst state of the system, we have $\Pr\{\phi \ge 0\} = 1$.

To present a recursive algorithm for reliability evaluation of the defined multi-state weighted k-out-of-n: G system, we first define the following notation:

- n : the number of components in the system
- M : the highest possible state of each component and system
- $w_{i,j}$: the utility of component i when it is in state j
- $p_{i,j}$: Pr{ Component i is in state j}
- $q_{i,j}$: Pr{ Component i is in a state below j}, $q_{i,j} = \sum_{l=0}^{j-1} p_{i,l}$.
- k_j : the minimum total utility required to ensure that the system is in state j or above.
- $R_j^I(k_j, n)$: probability for the system to be in state j or above based on the defined model I.

A recursive equation for evaluation of the state distribution of the system (Model I) is as follows:

$$R_j^I(k_j, i) = \sum_{r=0}^{M} p_{i,r} \cdot R_j^I(k_j - w_{i,r}, i-1). \tag{4.5}$$

The boundary conditions for this recursive equation are:

$$R_j^I(k, 0) = 0, \text{ when } 0 < k \le k_j,$$
$$R_j^I(k, i) = 1, \text{ when } i \ge 0 \text{ and } k \le 0.$$

Example 4.2: Consider a multi-state weighted k-out-of-n:G system with three components. Every component has three possible states: 0, 1, 2. Table 4.2 and Table 4.3

give the reliability distribution and the utility distribution of all the components. The utility of each component in state 0 is 1 instead of 0. Because in some situations, even the component works in the lowest state, it can still contribute some basic utilities to the system.

Table 4.2: Reliability distribution of the components, $p_{i,j}$

	$j=0$	$j=1$	$j=2$
$i=1$	0.1	0.2	0.7
$i=2$	0.4	0.2	0.4
$i=3$	0.3	0.5	0.2

Table 4.3: Utility distribution of the components, $w_{i,j}$

	$j=0$	$j=1$	$j=2$
$i=1$	1	2	3
$i=2$	1	3	4
$i=3$	1	3	5

In this example, $n=3$, $M=2$, $k_1=5$, and $k_2=10$. We can use Equation (4.5) to calculate the reliability of the system.

$$
\begin{aligned}
R_1^I(5,3) &= \sum_{r=0}^{2} p_{3,r} \cdot R_1^I(5-w_{3,r},3-1) \\
&= p_{3,0} \cdot R_1^I(5-1,2) + p_{3,1} \cdot R_1^I(5-3,2) + p_{3,2} \cdot R_1^I(5-5,2) \\
&= p_{3,0} \cdot R_1^I(4,2) + p_{3,1} \cdot R_1^I(2,2) + p_{3,2} \cdot R_1^I(0,2) \\
&= p_{3,0} \cdot R_1^I(4,2) + p_{3,1} + p_{3,2}
\end{aligned}
$$

$$R_1^I(4,2) = \sum_{r=0}^{2} p_{2,r} \cdot R_1^I(4 - w_{2,r}, 2-1)$$
$$= p_{2,0} \cdot R_1^I(4-1,1) + p_{2,1} \cdot R_1^I(4-3,1) + p_{2,2} \cdot R_1^I(4-4,1)$$
$$= p_{2,0} \cdot R_1^I(3,1) + p_{2,1} \cdot R_1^I(1,1) + p_{2,2} \cdot R_1^I(0,1)$$
$$= p_{2,0} \cdot p_{1,2} + p_{2,1} + p_{2,2} = 0.4 \cdot 0.7 + 0.2 + 0.4 = 0.88$$

$$R_1^I(5,3) = p_{3,0} \cdot R_1^I(4,2) + p_{3,1} + p_{3,2} = p_{3,0} \cdot 0.88 + p_{3,1} + p_{3,2}$$
$$= 0.3 * 0.88 + 0.5 + 0.2 = 0.964$$

$$R_2^I(10,3) = \sum_{r=0}^{2} p_{3,r} \cdot R_2^I(10 - w_{3,r}, 3-1)$$
$$= p_{3,0} \cdot R_2^I(10-1,2) + p_{3,1} \cdot R_2^I(10-3,2) + p_{3,2} \cdot R_2^I(10-5,2)$$
$$= p_{3,0} \cdot R_2^I(9,2) + p_{3,1} \cdot R_2^I(7,2) + p_{3,2} \cdot R_2^I(5,2)$$

$$R_2^I(5,2) = \sum_{r=0}^{2} p_{2,r} \cdot R_2^I(5 - w_{2,r}, 2-1)$$
$$= p_{2,0} \cdot R_2^I(5-1,1) + p_{2,1} \cdot R_2^I(5-3,1) + p_{2,2} \cdot R_2^I(5-4,1)$$
$$= p_{2,0} \cdot R_2^I(4,1) + p_{2,1} \cdot R_2^I(2,1) + p_{2,2} \cdot R_2^I(1,1)$$
$$= p_{2,0} \cdot 0 + p_{2,1} \cdot (p_{1,1} + p_{1,2}) + p_{2,2} = 0.2 \cdot (0.2 + 0.7) + 0.4 = 0.58$$

$$R_2^I(7,2) = p_{1,2} \cdot p_{2,2} = 0.7 * 0.4 = 0.28$$

$$R_2^I(9,2) = 0,$$

$$R_2^I(10,3) = p_{3,1} \cdot 0.28 + p_{3,2} \cdot 0.58 = 0.5 \cdot 0.28 + 0.2 \cdot 0.58 = 0.256$$

Thus the state distribution of the system is as follows:

$$\Pr(\phi \geq 0) = 1,$$
$$\Pr(\phi \geq 1) = 0.964,$$
$$\Pr(\phi \geq 2) = 0.256.$$
$$\Pr(\phi = 2) = 0.256,$$
$$\Pr(\phi = 1) = 0.964 - 0.256 = 0.708,$$
$$\Pr(\phi = 0) = 1 - 0.964 = 0.036.$$

We would also like to investigate the use of the UGF approach for reliability evaluation of multi-state weighted *k*-out-of-*n*:G systems. The UGF of each multi-state component is now given by:

$$U_i(z) = p_{i,0}z^{w_{i,0}} + p_{i,1}z^{w_{i,1}} + \cdots + p_{i,M}z^{w_{i,M}}. \tag{4.6}$$

To obtain the system UGF using the component UGFs, we still use the same operator Ω and δ_A as in the UGF method for binary weighted *k*-out-of-*n* systems.

Example 4.3: The multi-state weighted system studied in Example 4.2 is considered here. The UGF for the three components are as follow:

$$U_1(z) = 0.1z^1 + 0.2z^2 + 0.7z^3,$$
$$U_2(z) = 0.4z^1 + 0.2z^3 + 0.4z^4,$$
$$U_3(z) = 0.3z^1 + 0.5z^3 + 0.2z^5.$$

Based on the individual UGFs of the components, we can get the system UGF by using operator Ω as follows:

$$\begin{aligned} U_s(z) &= \Omega(U_1(z), U_2(z), U_3(z)) \\ &= \Omega((0.1z^1 + 0.2z^2 + 0.7z^3),(0.4z^1 + 0.2z^3 + 0.4z^4),(0.3z^1 + 0.5z^3 + 0.2z^5)) \\ &= 0.012z^3 + 0.02z^5 + 0.008z^7 + 0.006z^5 + 0.01z^7 + 0.004z^9 + 0.012z^6 + 0.02z^8 \\ &\quad + 0.008z^{10} + 0.024z^4 + 0.04z^6 + 0.016z^8 + 0.012z^6 + 0.02z^8 + 0.008z^{10} + 0.024z^7 \\ &\quad + 0.04z^9 + 0.016z^{11} + 0.084z^5 + 0.14z^7 + 0.056z^9 + 0.042z^7 + 0.07z^9 + 0.028z^{11} \\ &\quad + 0.084z^8 + 0.14z^{10} + 0.056z^{12} \end{aligned}$$

The equation given above is the output performance distribution of the multi-state weighted *k*-out-of-*n*: G system model I. From this equation, we can obtain the system state distribution for values of $k_1 = 5$ and $k_2 = 10$ using the operator δ_A, just as shown in Equation (4.4).

$$R_1^I(5,3) = \delta_4(U_s(z),5)$$
$$= 0.02+0.008+0.006+0.01+0.004+0.012+0.02$$
$$+0.008+0.04+0.016+0.012+0.02+0.008+0.024+0.04+0.016+0.084$$
$$+0.14+0.056+0.042+0.07+0.028+0.084+0.14+0.056$$
$$= 0.964,$$

$$R_2^I(10,3) = \delta_4(U_s(z),10) = 0.008+0.008+0.016+0.028+0.14+0.056$$
$$= 0.256.$$

Actually, the above process can be simplified by collecting the like terms in $U_s(z)$ and reducing its length from 27 terms to 10 terms. Of course, this simplification process can also be applied to all intermediate polynomials to calculate $U_s(z)$ such as the result of $\Omega(U_1(z),U_2(z))$ based on the properties of operator Ω .

$$U_s(z) = 0.012z^3+0.02z^5+0.008z^7+0.006z^5+0.01z^7+0.004z^9+0.012z^6+0.02z^8$$
$$+0.008z^{10}+0.024z^4+0.04z^6+0.016z^8+0.012z^6+0.02z^8+0.008z^{10}+0.024z^7$$
$$+0.04z^9+0.016z^{11}+0.084z^5+0.14z^7+0.056z^9+0.042z^7+0.07z^9+0.028z^{11}$$
$$+0.084z^8+0.14z^{10}+0.056z^{12}$$
$$= 0.012z^3+0.024z^4+0.011z^5+0.0106z^6+0.182z^7+0.2z^8+0.1z^9+0.184z^{10}$$
$$+0.016z^{11}+0.056z^{12}$$

In this way, the calculation time can be reduced a lot. However, it depends on how many like terms exist. Sometimes, there are few like terms in $U_s(z)$ especially when the utility vectors of components are different decimal numbers. We can see it later in the comparison examples.

The distribution of the system state in this example is the same as obtained in Example 4.2. For the multi-state weighted k-out-of-n: G system model presented in this section, we also compared the performances of the recursive method and the UGF approach. The CPU times required by the two methods were obtained as the values M and n when

only one of them changed, and they are compared in two situations. One is when the utilities of the component are random integer numbers from a given range. The other is when the component utilities are random decimal numbers from a given range. The integer numbers for the component utility are randomly selected from [1,14]. The decimal numbers for the component utility are randomly selected from [20,50]. The probability of component i in state j is randomly selected from [0,1] and meets the requirement of $\sum_{j=0}^{M} p_{ij} = 1$. The reason to set up the experiments like this is to show the effects of two situations: with lots of like terms in the intermediate polynomials to calculate $U_s(z)$ and with few like terms in the intermediate polynomials to calculate $U_s(z)$. These comparisons are shown in Table 4.4, Table 4.5, Table 4.6, and Table 4.7.

Based on the CPU times required by the two methods for various values of the parameters M and n of the multi-state weighted k-out-of-n: G model, we have the following observations. As the system size n increased, the CPU time required by the UGF approach increased much faster than that by the recursive method. As the number of component states M increased, the CPU time required by the UGF approach increased much faster than that by the recursive method. In addition, when there are many like terms in the process of calculating $U_s(z)$, collecting the like terms can reduce the calculation time a lot; when there are seldom like terms in the process of calculating $U_s(z)$, collecting the like terms may make the calculation slower, because collecting the like terms itself also spends time. Generally speaking, the recursive approach is more efficient than the UGF approach for multi-state weighted k-out-of-n:G system: Model I performance evaluation.

Table 4.4: Comparison of the CPU times of the two methods when only M changes with random integer number for the component utility ($k_1 = 200$, $n = 5$)

The M Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0000	0.0000	0.0025
4	0.0460	0.0160	0.0028
5	0.1220	0.0460	0.0038
6	0.3210	0.1830	0.0048
7	0.6170	0.7110	0.0049
8	1.1120	3.2130	0.0061
9	2.0660	12.0400	0.0080
10	3.3810	43.2040	0.0101
11	5.4220	113.0950	0.0131
12	8.3380	256.1410	0.0151

Table 4.5: Comparison of the CPU times of the two methods when only n changes with random integer number for the component utility ($k_1 = 200$, $M = 5$)

The n Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0000	0.0000	0.0023
4	0.0160	0.0000	0.0028
5	0.1220	0.0460	0.0038
6	0.6070	0.9190	0.0048
7	3.0270	44.3890	0.0058
8	14.9210	1566.9123	0.0073
9	75.5460	More than one hour	0.0091
10	225.0440		0.0114
11	1872.3541		0.0138

Table 4.6: Comparison of the CPU times of the two methods when only n changes with random decimal number for the component utility ($k_1 = 200$, $M = 5$)

The n Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0000	0.0000	0.0029
4	0.0160	0.0000	0.0295
5	0.1220	0.0460	0.3044
6	0.6070	0.9190	7.6404
7	3.0270	44.3890	263.7771
8	14.9210	1566.9123	More than one hour
9	75.5460	More than one hour	

Table 4.7: Comparison of the CPU times of the two methods when only M changes with random decimal number for the component utility ($k_1 = 200$, $n = 5$)

The M Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0000	0.0000	0.0037
4	0.0460	0.0160	0.0368
5	0.1220	0.0460	0.3044
6	0.3210	0.1830	1.8191
7	0.6170	0.7110	8.3769
8	1.1120	3.2130	32.2615
9	2.0660	12.0400	139.3600
10	3.3810	43.2040	437.2028
11	5.4220	113.0950	1116.4360
12	8.3380	256.1410	2668.0371

4.3 Multi-state weighted k-out-of-n:G system: model II

Huang et al. [4] proposed the general multi-state k-out-of-n system model in 2000. In their definition, for the system state to be not lower than a given value j, the number of components whose states are not lower than j must be at least k_j, a pre-specified value. In this definition, the components whose states are below j do not make any contribution for the system to be in state j or above. Based on this idea, we define a new model of the multi-state weighted k-out-of-n: G system. In the proposed definition, for the system to be in state j or above, the sum of the utilities of only those components whose states are in state j or above must be not less than k_j, a prespecified value. The difference between Model I presented in the previous section and Model II presented in this section is whether the components whose states are below j are making any contribution for the system to be in state j or above. The formal definition of Model II is given below.

Definition 4.2: The system is in state j or above if the sum of the utilities of the components whose states are in state j or above is greater than or equal to k_j. Let ϕ be the structure function of the system and W_j be the sum of the utilities of the components whose states are j or above. We then have $\Pr\{\phi \geq j\} = \Pr\{W_j \geq k_j\}$.

Let $R_j^{II}(k,n)$ denote the probability for the n component system to have a sum of useful utilities of at least k_j when one is evaluating the probability for the system to be in state j or above. The following recursive algorithm is proposed for evaluation of the performance distribution of the system based on Definition 4.2.

$$R_j^{II}(k_j,n) = q_{n,j} \cdot R_j^{II}(k_j,n-1) + \sum_{r=j}^{M} p_{n,r} \cdot R_j^{II}(k_j - w_{n,r}, n-1). \qquad (4.7)$$

The boundary conditions for equation (4.7) are:

$$R_j^{II}(j,0) = 0, \text{ for } j = 1,2,3,\cdots,k_j,$$
$$R_j^{II}(k,i) = 1, \text{ for } k \leq 0 \text{ and } i = 0,1,2,\cdots,n$$

Example 4.4: We consider the same set of components used in Example 4.2. However, the system state is determined based on Definition 4.2. Thus, we have $n = 3$, $M = 2$, $k_1 = 5$, and $k_2 = 10$. Equation (4.7) is used below to calculate the state distribution of the system.

$$R_1^{II}(5,3) = q_{3,1} \times R_1^{II}(5,2) + \sum_{r=1}^{2} p_{3,r} \times R_1^{II}(5 - w_{3,r}, 3-1)$$
$$= q_{3,1} \times R_1^{II}(5,2) + p_{3,1} \times R_1^{II}(5-3,2) + p_{3,2} \times R_1^{II}(5-5,2)$$
$$= 0.3 \times R_1^{II}(5,2) + 0.5 \times R_1^{II}(2,2) + 0.2 \times R_1^{II}(0,2)$$
$$= 0.3 \times R_1^{II}(5,2) + 0.5 \times R_1^{II}(2,2) + 0.2,$$

$$R_1^{II}(5,2) = q_{2,1} \times R_1^{II}(5,1) + \sum_{r=1}^{2} p_{2,r} \times R_1^{II}(5 - w_{2,r}, 2-1)$$
$$= q_{2,1} \times R_1^{II}(5,1) + p_{2,1} \times R_1^{II}(2,1) + p_{2,2} \times R_1^{II}(1,1)$$
$$= p_{2,1} \times (0.2+0.7) + p_{2,2} \times (0.2+0.7)$$
$$= (0.2+0.4) \times 0.9$$
$$= 0.54,$$

$$R_1^{II}(2,2) = q_{2,1} \times R_1^{II}(2,1) + \sum_{r=1}^{2} p_{2,r} \times R_1^{II}(2 - w_{2,r}, 2-1)$$
$$= q_{2,1} \times R_1^{II}(2,1) + p_{2,1} \times R_1^{II}(-1,1) + p_{2,2} \times R_1^{II}(-2,1)$$
$$= 0.4 \times (0.2+0.7) + 0.2 + 0.4$$
$$= 0.96,$$

$$R_1^{II}(5,3) = 0.3 \times R_1^{II}(5,2) + 0.5 \times R_1^{II}(2,2) + 0.2$$
$$= 0.3 \times 0.54 + 0.5 \times 0.96 + 0.2 = 0.842,$$

$$R_2^{II}(10,3) = q_{3,2} \times R_2^{II}(10,2) + \sum_{r=2}^{2} p_{3,r} \times R_2^{II}(10 - w_{3,r}, 3-1)$$
$$= q_{3,2} \times R_2^{II}(10,2) + p_{3,2} \times R_2^{II}(10-5,2)$$
$$= q_{3,2} \times R_2^{II}(10,2) + p_{3,2} \times R_2^{II}(5,2),$$
$$R_2^{II}(10,2) = 0,$$

$$R_2^{II}(5,2) = q_{2,2} \times R_2^{II}(5,1) + \sum_{r=2}^{2} p_{2,r} \times R_2^{II}(5 - w_{2,r}, 2-1)$$
$$= p_{2,0} \times R_2^{II}(5,1) + p_{2,2} \times R_2^{II}(5-4,1)$$
$$= p_{2,0} \times R_2^{II}(5,1) + p_{2,2} \times R_2^{II}(1,1)$$
$$= p_{2,0} \times 0 + p_{2,2} \times p_{1,2}$$
$$= 0.28,$$

$$R_2^{II}(10,3) = q_{3,2} \times R_2^{II}(10,2) + p_{3,2} \times R_2^{II}(5,2) = p_{3,2} \times 0.28$$
$$= 0.2 \times 0.28 = 0.056.$$

Furthermore, extending the UGF from Model I to Model II, we only need to change the UGF for the individual component as the following form:

$$U_i(z) = q_{i,j} z^0 + p_{i,j} z^{w_{i,j}} + \cdots p_{i,k} z^{w_{i,k}} + \cdots + p_{i,M} z^{w_{i,M}}. \qquad (4.8)$$

For the system in Example 4.4, when we calculate the probability that the system is in state 1 or above, the UGF for the three components should be written as follow:

$$U_1(z) = 0.1z^0 + 0.2z^2 + 0.7z^3,$$
$$U_2(z) = 0.4z^0 + 0.2z^3 + 0.4z^4,$$
$$U_3(z) = 0.3z^0 + 0.5z^3 + 0.2z^5$$

Based on the individual UGF of the components, we can get the system UGF by using operator Ω:

$$U_s(z) = \Omega(U_1(z), U_2(z), U_3(z))$$
$$= 0.012z^0 + 0.024z^2 + 0.11z^3 + 0.012z^4 + 0.06z^5 + 0.216z^6 + 0.12z^7$$
$$+ 0.08z^8 + 0.118z^9 + 0.148z^{10} + 0.044z^{11} + 0.056z^{12}$$

The above form is the MS weighted *k*-out-of-*n* Model II OPD. We can obtain the system reliability for the arbitrary $k_1 = 5$ based on this form using the operator δ_A:

$$R_1^{II}(5,3) = \delta_A(U_s(z),5) = 0.842$$

When we calculate the probability that the system is in state 2 or above, the UGF for the three components should be written as follow:

$$U_1(z) = 0.3z^0 + 0.7z^3,$$
$$U_2(z) = 0.6z^0 + 0.4z^4,$$
$$U_3(z) = 0.8z^0 + 0.2z^5$$

Based on the individual UGF of the components, we can get the system UGF by using operator Ω:

$$\begin{aligned} U_s(z) &= \Omega(U_1(z),U_2(z),U_3(z)) \\ &= 0.144z^0 + 0.336z^3 + 0.096z^4 + 0.036z^5 + 0.224z^7 + 0.084z^8 + 0.024z^9 + 0.056z^{12} \end{aligned}$$

We can obtain the system reliability for the arbitrary $k_2 = 10$ based on this form using the operator δ_A:

$$R_2^{II}(10,3) = \delta_A(U_s(z),10) = 0.056$$

So we get the same results as using the recursive method.

In summary, we have the state distribution of the system as follows:

$$\Pr(\phi \geq 1) = R_1^{II}(5,3) = 0.842,$$
$$\Pr(\phi \geq 0) = 1,$$
$$\Pr(\phi = 2) = 0.056,$$
$$\Pr(\phi = 1) = 0.842 - 0.056 = 0.786,$$
$$\Pr(\phi = 0) = 1 - 0.842 = 0.158.$$

Given $j = 3$, using the same method in Section 4.2 to compare the two approaches. The results are shown in Table 4.8, Table 4.9, Table 4.10, and Table 4.11. From these results, we can say generally speaking, the recursive approach is more efficient than the UGF approach for multi-state weighted k-out-of-n:G system: Model II performance evaluation.

Table 4.8: Comparison of the CPU times of the two methods when only M changes with random integer number for the component utility ($k_1 = 200$, $n = 5$)

The M Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0160	0.0000	0.0030
4	0.0160	0.0000	0.0037
5	0.0470	0.0150	0.0039
6	0.1250	0.0470	0.0045
7	0.2970	0.1880	0.0051
8	0.6410	0.7650	0.0061
9	1.2340	2.6870	0.0075
10	2.1400	12.1060	0.0095
11	3.5460	42.2850	0.0121
12	5.6230	111.7170	0.0138

Table 4.9: Comparison of the CPU times of the two methods when only n changes with random integer number for the component utility ($k_1 = 200$, $M = 5$)

The n Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0000	0.0000	0.0029
4	0.0150	0.0000	0.0038
5	0.0470	0.0150	0.0039
6	0.1880	0.0940	0.0049
7	0.7190	1.2810	0.0060
8	3.0160	41.4560	0.0073
9	11.5930	885.1180	0.0091
10	46.7640	More than one hour	0.0109

Table 4.10: Comparison of the CPU times of the two methods when only M changes with random decimal number for the component utility ($k_1 = 200$, $n = 5$)

The M Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0160	0.0000	0.0033
4	0.0160	0.0000	0.0065
5	0.0470	0.0150	0.0396
6	0.1250	0.0470	0.3076
7	0.2970	0.1880	1.8255
8	0.6410	0.7650	8.3420
9	1.2340	2.6870	31.8077
10	2.1400	12.1060	136.6563
11	3.5460	42.2850	404.5117
12	5.6230	111.7170	1142.2968

Table 4.11: Comparison of the CPU times of the two methods when only n changes with decimal decimal number for the component utility ($k_1 = 200, M = 5$)

The n Value	CPU (Seconds) Recursive Method	CPU (Seconds) UGF method without collecting like terms	CPU (Seconds) UGF method with collecting like terms
3	0.0000	0.0000	0.0030
4	0.0150	0.0000	0.0059
5	0.0470	0.0150	0.0396
6	0.1880	0.0940	0.5501
7	0.7190	1.2810	8.7883
8	3.0160	41.4560	190.7333
9	11.5930	885.1180	3535.5925
10	46.7640	More than one hour	More than one hour

Examples 4.3 and 4.4 use the same set of components. The only difference between these two examples is that what components' utilities are used in determining the state of the system. The state distributions obtained in these two examples are given in Table 4.12.

Table 4.12: Comparison of system state distributions of Model I and Model II

	Model I	Model II
$\Pr(\phi \geq 2)$:	0.256	0.056
$\Pr(\phi \geq 1)$:	0.964	0.842
$\Pr(\phi \geq 0)$:	1.000	1.000

From Table 4.12, it is apparent that the probability for the Model II system to be not less than a specific state is less than that for the Model I system. Model I may be applied whenever the contribution of every component is useful no matter how bad or good a component is. Model II is applicable when components in bad states cannot make any contribution for operation of high system states.

4.4 Conclusions

In this chapter we have proposed two definitions of multi-state weighted k-out-of-n:G system model. They may be applied in different situations when the contributions of relatively weak components may and may not be useful. Recursive algorithms are provided for evaluation of system distribution under both definitions. The UGF approach is compared with recursive methods for the binary weighted k-out-of-n:G system defined by Wu and Chen [18] and the proposed two kinds of multi-state weighted k-out-of-n:G systems. It is found that recursive methods are generally more efficient than the UGF approach.

In the design process, we need to evaluate system reliability repetitively. Using the more efficient reliability evaluation method in the MS weighted system optimal reliability design is very important. Based on the research in this chapter, we can select more efficient method in the MS weighted k-out-of-n system optimal reliability design which will be presented in the following chapter.

References:

[1]. Y. Chen and Q. Y. Yang. Reliability of two-stage weighted-k-out-of-n systems with components in common. IEEE Transactions on Reliability, 54 (3): 431-440, 2005.

[2]. B. Gnedenko and I. Ushakov. Probabilistic Reliability Engineering. John Wiley and Sons, Inc, NY/Chichester/Brisbane, 1995.

[3]. Y. Higashiyama. A factored reliability formula for weighted-k-out-of-n system. Asia-Pacific Journal of Operational Research, 18 (1): 61-66, 2001.

[4]. J. Huang, M. J. Zuo and Y. H. Wu. Generalized multi-state k-out-of-n:G system. IEEE Transaction on Reliability, R-49 (1): 105-111, 2000.

[5]. G. Levitin. Optimal series-parallel topology of multi-state system with two failure modes. Reliability Engineering and System Safety, 77 (1): 93-107, 2002.

[6]. G. Levitin and A. Lisnianski. Structure optimization of power system with bridge topology, Electric Power Systems Research, 45 (3): 201-208, 1998.

[7]. G. Levitin and A. Lisnianski. Reliability optimization for weighted voting system. Reliability Engineering and System Safety, 71 (2): 131-138, 2001.

[8]. G. Levitin and A. Lisnianski. Structure optimization of multi-state system with two failure modes. Reliability Engineering and System Safety, 72 (1): 75-89, 2001.

[9]. G. Levitin, A. Lisnianski and D. Elmakis. Structure optimization of power system with different redundant elements. Electric Power Systems Research, 43 (1): 19-27, 1997.

[10]. G. Levitin, A. Lisnianski, H. Ben-Haim and D. Elmakis. Redundancy optimization for series-parallel multi-state systems. IEEE Transactions on Reliability, 47 (2): 165-172, 1998.

[11]. W. Li and M.J. Zuo. Reliability evaluation of multi-state weighted *k*-out-of-*n* systems. Reliability Engineering and System Safety, 93(1): 161-168, 2008.

[12]. A. Lisnianski, G. Levitin and H. Ben Haim. Structure optimization of multi-state system with time redundancy. Reliability Engineering and System Safety, 67 (2): 103-112, 2000.

[13]. A. Lisnianski, G. Levitin, H. Ben-Haim and D. Elmakis. Power system structure optimization subject to reliability constraints. Electric Power Systems Research, 39 (2): 145-152, 1996.

[14]. P.D.T. O'Connor. Practical Reliability Engineering. John Wiley & Sons, 2002.

[15]. I. Ushakov. Universal generating function. Soviet Journal Computer and System Science, 24 (5): 118-129, 1986.

[16]. I. Ushakov. Optimal standby problem and a universal generating function. Soviet Journal Computer and System Science, 25 (4): 61-73, 1987.

[17]. I. Ushakov. The method of generalized generating sequences. European Journal of Operational Research, 125 (2): 316-323, 2000.

[18]. J. S. Wu and R. J. Chen. An algorithm for computing the reliability of a weighted *k*-out-of-*n* system. IEEE Transactions on Reliability, R-43(2): 327-328, 1994.

Chapter 5

Optimal Design of Binary Weighted *k*-out-of-*n* Systems

In this chapter, based on the reliability evaluation methods introduced in Chapter 4, we consider the optimal design of the binary weighted *k*-out-of-*n* system. The binary weighted *k*-out-of-*n*:G system works if and only if the total utility of all working components is at least *k*. In the design process, we need to evaluate system reliability repetitively. The universal generating function (UGF) approach is used for this purpose when the system size is small or moderate, and when the size of the system is large, the more efficient recursive approach is used. Two optimization models are formulated. One is to minimize the expected total cost at the same time guarantee the system reliability bigger than a pre-specified value, and the other is to maximize the system reliability with the constraints on total system cost. Genetic Algorithms (GA) and Tabu Search (TS) methods are used to solve the resulting optimization models. Since the key to a good TS algorithm is usually quite problem-specific policies and memory structures for deciding what a move is and how long moves are tabu after leading to a local minimum, there is no existing general TS tool available. So, much more details of the TS approach used in this chapter are given than GA approach. The results obtained with these two methods are compared. The results show that both are powerful tools to solve these kinds of problems, but TS is more efficient. A simpler version of the materials in this chapter has been published in [22], and an extended version has been submitted to the International Journal of Reliability, Quality and Safety Engineering [23].

5.1 Introduction

In a binary k-out-of-n: G system, the system works if and only if at least k components work. In a k-out-of-n:F system, the system fails if and only if at least k components fail. Wu and Chen [34] generalized these k-out-of-n system models into the weighted k-out-of-n models. In a weighted k-out-of-n:G system, component i carries a weight of w_i, $w_i > 0$ for $i = 1, 2, \ldots, n$. The total weight of all components is w, $w = \sum_{i=1}^{n} w_i$. The system works if and only if the total weight of working components is at least k, a pre-specified value. Since k is a weight, it may be larger than n because they have different measuring units. Such a weighted k-out-of-n:G system is equivalent to a weighted $(w-k+1)$-out-of-n:F system wherein the system fails if and only if the total weight of failed components is at least $w-k+1$. With these generalizations, the k-out-of-n system models are special cases of the weighted k-out-of-n system models since each component in a k-out-of-n system has a weight of one.

Actually the "weight" Wu and Chen mentioned in their paper means the contribution of the component. The component with higher contribution to the system has higher "weight". The component with lower contribution to the system has lower "weight". For example, a jet plane usually has several engines. The engine with higher drive has higher "weight". The engine with lower drive has lower "weight". In this chapter, in order to distinguish the "weight" from "physical weight" clearly, we use "utility" to substitute the name of "weight" and use notation u_i to substitute w_i in above.

To evaluate the reliability of a binary weighted k-out-of-n:G system, Wu and Chen [34] provided a recursive algorithm. Higashiyama [3] provided a method to express the system

reliability of a weighted k-out-of-n system in fewer terms than that with the algorithm by Wu and Chen [34]. However, the time complexity and space complexity of these two methods reported in [3] and [34] are the same. The most recently reported study on the binary weighted k-out-of-n systems is by Chen and Yang [3]. In [3], the one-stage weighted k-out-of-n model was extended to the two-stage weighted k-out-of-n model with components in common among the stages.

The concept of universal generating function (UGF) was introduced by Ushakov [32] in 1986. Detailed mathematical foundations of the UGF method were presented in Gnedenko and Uskakov [9], Reinshke and Ushakov [28], and Ushakov [13]. In 1996, Lisnianski et al. [20] applied this method to power system reliability analysis. Since UGF is a general technique, one needs to develop specific operators for different kinds of system structures. In a series of research work by Levitin and Lisnianski [18], [20], [15], [18], [16], [17] and [13], several operators of the UGF method were provided for performance evaluation of several multi-state system structures including series, parallel, series-parallel, and the bridge. In our research work [24], we applied UGF method to not only the multi-state weighted k-out-of-n system, but also the binary weighted k-out-of-n system reliability evaluation.

Thus to evaluate the reliability a weighted k-out-of-n: G system, we may use either the recursive algorithm by Wu and Chen [34] or the UGF approach. To compare the computational complexity of these two approaches [24], we get the follow results. When the size of the system is very small ($n \le 6$), the required computation CPU times of the two methods are about the same; when the size of the system is moderate ($7 \le n \le 13$), the UGF method seems to be more efficient, and when the size of the system is large ($n \ge 14$), the

recursive approach is more efficient. In addition, when k is the only parameter that may change, that is, all other parameters such as system size n, component reliabilities, and component weights are fixed, the UGF approach is more efficient than the recursive approach.

In solving an optimal design problem, we need to evaluate system reliability repetitively. Based on the comparison results in terms of the efficiency of the two approaches, the more efficient one will be used in optimal system design.

The multi-state system mentioned above is a more general system than the binary system. In the multi-state context, a component may be in more than two different states, varying from perfect functioning to complete failure. When in a different state, it may make a different contribution to the system. In the binary context, a component has only two states: perfect working and failed. The binary system is a special case of the multi-state system. In this chapter, we only study the binary weighted k-out-of-n system.

5.2 Design models

In optimal design of k-out-of-n systems, there are reported studies for determining the optimal system size n [26], finding the optimal value k [27], and determining the optimal values n and k simultaneously [28]. The commonly used form of the objective function to be minimized is $cn + d(1 - R_s)$, where c is the cost of each component, d is the cost of system failure, and R_s is system reliability for some time period.. This objective function is interpreted as the "expected total cost = the costs of all components + the expected cost of system failure."

In this study, we focus on the optimal design of the weighted k-out-of-n system. We choose to minimize the expected total cost of the system subject to a minimum system reliability requirement. The expected total cost includes the costs of all components and the expected cost of system failure as described before. The decision variables are not n, k, or both. All these are fixed. We are to select n components from a set of available components with possibly different reliabilities and weights. The optimization problem which we call problem **P1** can be formulated as follow.

Problem P1:

Minimize:

$$C_s = \sum_{i=1}^{n} c_i + (1 - R_s) \cdot C_f \tag{5.1}$$

Subject to:

$$R_s \geq R^*$$

Notation:

- c_i : the design and manufacturing cost of component i
- C_f : the cost of system failure (includes repair cost and production loss)
- C_s : expected total cost
- R_s : system reliability
- R^* : minimum system reliability required
- n : the number of components in the system.

R_s is a function of component reliabilities. As mentioned before, either the recursive

algorithm by Wu and Chen [34] or the UGF approach can be used to calculate the system reliability R_s. The component reliabilities are given in the database. The component reliabilities and system reliability are all probability numbers. According to the definition of reliability, it means the component/system can perform its operating function under the pre-defined situation and during the pre-defined period of time. The design model we studied here is the component selection model. There are discrete component choices with known characters (cost, utility, reliability). The objective is to determine which component to use. The decision variables in this optimization problem are the component ID numbers which will be selected from a database of available components. In fact, this is a component selection problem which has been studied in recent years in optimal design of systems of various structures. In 1990, Shen and Xie [30] studied the component selection problem for a parallel system to maximize system reliability. In 2003, Coit [5] studied the redundancy allocation problem with discrete component choices for multi-state series-parallel systems. The objective function was to maximize system reliability subject to constraints on system cost and system weight. In 2004, Ramirez-Marquez and Coit [28] continued the study of the redundancy allocation problem with discrete component choices for multi-state series-parallel systems. The objective function was to minimize system cost subject to system reliability requirement. In a series of research work by Levitin and Lisnianski [18], [20], [15], [18], [16], [17] and [13], they studied the following component selection problems:

1. maximizing system reliability without considering costs at all,
2. minimizing system cost subject to requirement on system reliability.

The optimization model that we have formulated here is for optimal design of the weighted k-out-of-n systems. Both system reliability and system costs will be considered. The components do not have to be identical. When we select components, we will consider the component weights, component costs, and component reliabilities simultaneously. In addition, our objective function is the expected total cost of the system, not only the costs of the components.

Furthermore, in problem **P1** we try to find the optimal solution that minimize the expected total cost with the constraint that the system reliability is bigger than a pre-specified value. There is another problem we call it problem **P2** which is to maximize the system reliability with the constraints on total component cost and physical weight. A similar problem has been studied in [4]. However, in [4] they considered the system configured by connecting binary k-out-of-n subsystem (not weighted k-out-of-n subsystems) in series. There is only one component choice used for each subsystem. That is to say, in each k-out-of-n subsystem all the components are the same. Here, we consider the optimal design of weighted k-out-of-n systems, and all the components in the system can be different. The decision variables in problem **P2** are still the choices for component ID numbers. The objective is to select the component choices to maximize the system reliability with the constraints of components cost and physical weight. The problem **P2** is formulated as:

Problem P2:

Maximize:

$$R_s \tag{5.2}$$

Subject to:

$$\sum_{i=1}^{n} c_i + (1 - R_s) \cdot C_f \leq C^*$$

Notation:

- R_s : the system reliability
- c_i : the design and manufacturing cost of component i
- C_f : the cost of system failure (includes repair cost and production loss)
- C^* : the upper limit for the system expected total cost
- n : the number of component

5.3 Solution approaches and example results

Reported optimization approaches for solving reliability based design problems include dynamic programming, integer programming, mixed integer programming, non-linear programming, heuristics, and metaheuristics. A literature survey on these approaches was provided by Kuo and Prasad [12].

The family of metaheuristic optimization techniques includes simulated annealing, Tabu search (TS), Genetic Algorithm (GA), Evolutionary Strategies, Ant Colony, Immune Algorithm, Swarm Optimization and so on. The applications of some of these optimization techniques can be found in [14]. These methods have been found to be more powerful for solving large-scale and complex optimization problems. Especially, they perform much better than traditional optimization algorithms in finding global optimal solutions.

GA has been demonstrated to converge to global optimal solutions for many diverse difficult problems, although optimality cannot be guaranteed. TS is also very useful for

solving large complex optimization problems. Its salient feature is the use of memory (information about previous solutions) to guide the search beyond local optimality. There is no fixed sequence of operations in TS and its implementation is problem-specific. Actually the key to a good TS algorithm is usually quite problem-specific policies and memory structures for deciding what a move is and how long moves are tabu after leading to a local minimum. A simple TS which uses only short-term memory is easy to implement. Usually such methods yield good solutions when attributes, tabu-tenure, and aspiration-criteria are appropriately defined. In reference [12], the authors strongly recommended TS for solving complicated redundancy allocation problems. In reference [11], TS was demonstrated on numerous variations of three different problems and the solutions obtained with TS were compared to the solutions obtained with integer programming and GA. The comparison results confirmed that tabu search was very useful for solving large complex optimization problems that are very difficult or impossible to solve with mathematical programming techniques. When compared to GA, TS provided superior performance in terms of best solutions found and reduced variability and offered the potential of greater efficiency [11]. Besides the reliability optimization problems, TS has also demonstrated its advantages in other optimization problems. For example, in Morley's transport network optimization research [25], he found that TS is more efficient than GA.

In this work, we will use both TS and GA to solve the optimal design problem of a weighted k-out-of-n system and compare their performances. In the following, we will first provide an introduction of the TS approach.

TS uses a procedure to guide local search methods to overcome local optimality and

achieve global optimality or near-global optimality. Starting from an initial solution, the method explores the solution space and moves to the next solution in the neighborhood at each iteration. This next solution may not be a better solution and this allows the method to escape from a possibly local optimum and move to other regions of the search space. To avoid cycling, a specially designed memory mechanism, known as the tabu list, is used to store certain previously visited solutions or certain attributes of them. In particular, the status of a tabu move may also be overruled and made possible right away if a certain aspiration criterion is met. For a more comprehensive description of TS, readers can refer to reference [6].

The following lists the main elements of the TS approach:

1. Initial solution:

An initial solution needs to be generated. The method is not very sensitive to the starting solution. For our component selection problem, the solution includes n component ID numbers.

2. Moves used:

The two commonly used types of moves are the Swap Move and the Insert Move [8]. A Swap Move is to arbitrarily select two elements in a solution and switch their positions. An Insert Move is to arbitrarily select two elements in a solution, pick up one element and insert it in front of the other element. Neither of these two moves is applicable for optimal design of the weighted k-out-of-n system. The reason is that both of these moves change the positions of the elements in a solution and this does not change either the cost or the reliability of the weighted k-out-of-n system.

Based on the specific property of our problem, we define the following two types of moves. The first type of move is to change the ID number of a particular component type by adding or subtracting one. This move is considered for each element in the current solution. The second type of move simultaneously adds a component type to and drops another component type from the current solution simultaneously. All possibilities are enumerated for this second type of move. These two types of moves are preformed independently on the current solution and the best feasible solution among them will be selected. We use Figure 5.1 and Figure 5.2 to illustrate how they work. In Figure 5.1, we subtract 1 from the second component and this changes the second component type from 2 to 1. In Figure 5.2, we add a component type 4 and simultaneously drop component type 3 from the solution.

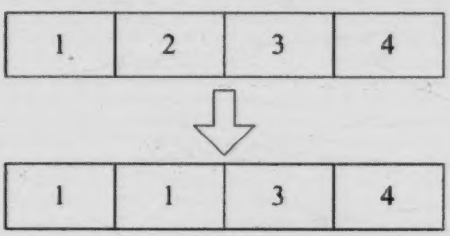


Figure 5.1: Type 1 move

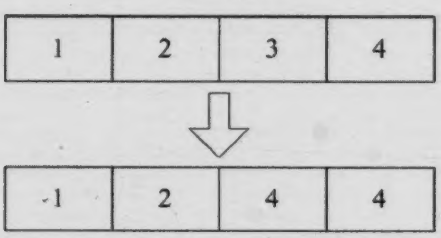


Figure 5.2: Type 2 move

3. Tabu list:

The most critical step of the TS approach is embedded in its short-term memory process. A tabu list (short term memory) is to record a limited number of attributes of solutions

(moves, selections, assignments, etc) which will be discouraged in order to prevent revisiting a visited solution. This will ensure the method to focus on the unvisited areas of the solution space. Without such restrictions, the method could take a so called "best" move away from a local optimum, and then conceivably at the next step, fall back into the local optimum by taking the best move available at that point. In general, the tabu restrictions are intended to prevent such cycling behavior and more broadly to induce the search to follow a new trajectory if cycling in a narrower sense occurs.

In several reported applications of the TS approach, the most commonly used length of the tabu list fell in the interval from 5 to 12, with 7 representing a highly effective value [98]. In this chapter, we will select 7 as the length of the tabu list.

4. Aspiration criterion:

In general, if after a move, the attributes of the new solution is in the tabu list, we usually drop this new solution. However, if the objective function value of this new solution is better than the best value found so far, we keep it in spite of its status of being in the tabu list.

5. Stopping criterion:

If the number of iterations reaches a specified number, or evidence can be given that an optimum solution has been obtained, or from the current solution, we cannot find any more feasible solutions, the optimization process terminates. In our numerical examples which are given later, the program always stops when the optimum solution has been obtained.

6. The search procedure:

The tabu search process for our problem is shown in Figure 5.3.

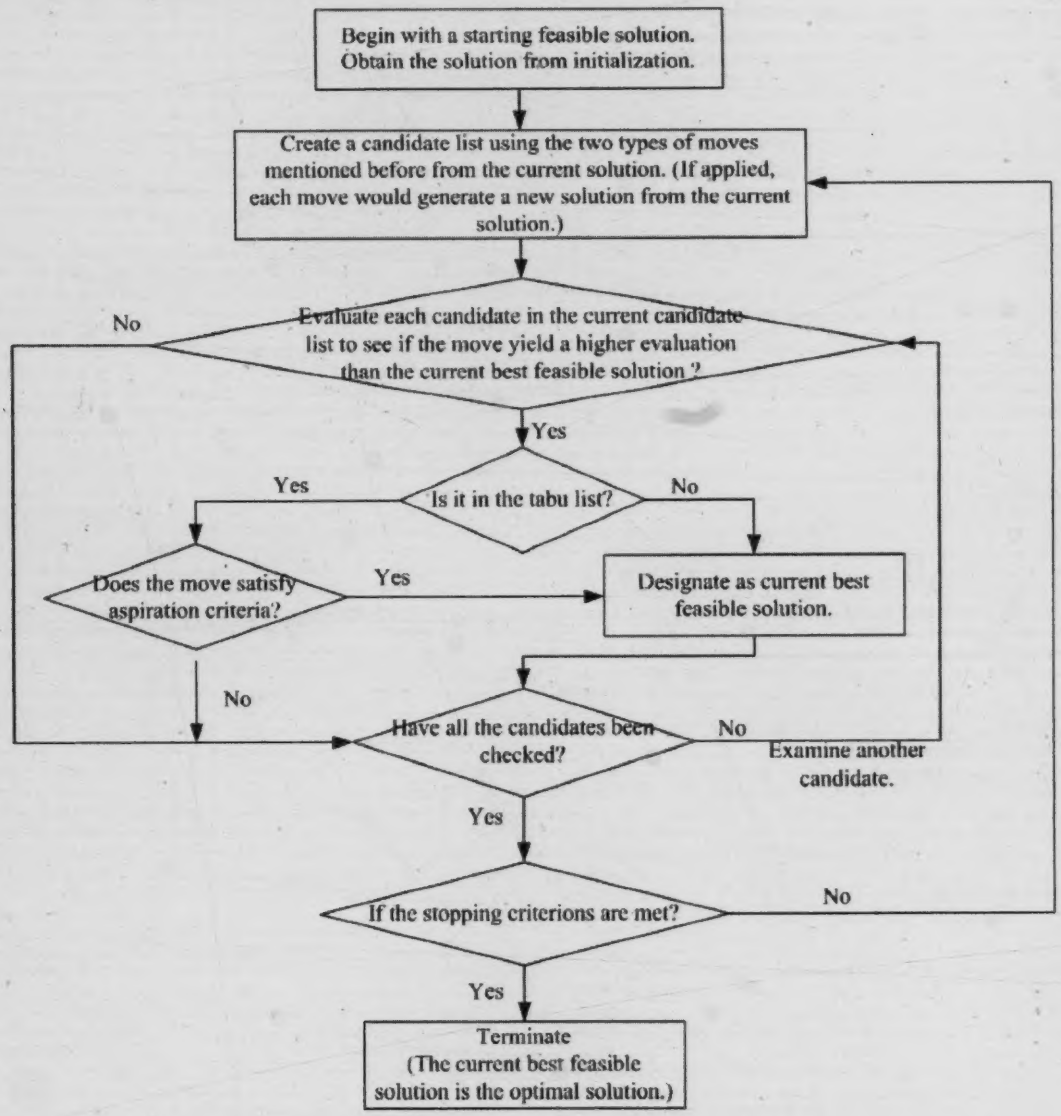

Begin with a starting feasible solution. Obtain the solution from initialization.
Create a candidate list using the two types of moves mentioned before from the current solution. (If applied, each move would generate a new solution from the current solution.)
Evaluate each candidate in the current candidate list to see if the move yield a higher evaluation than the current best feasible solution ?
No
Yes
Is it in the tabu list?
Yes
No
Does the move satisfy aspiration criteria?
Yes
Designate as current best feasible solution.
No
Have all the candidates been checked?
No
Examine another candidate.
Yes
If the stopping criterions are met?
No
Yes
Terminate (The current best feasible solution is the optimal solution.)

Figure 5.3: The flow chart of the tabu search procedure

In this study, we will use both GA and TS to solve the optimization problem in design of weighted k-out-of-n systems and compare their results. In order to make the calculation easier, we change the form of the optimization problem **P1** as follows using a penalty

function:

$$C_s = \sum_{i=1}^{n} c_i + (1 - R_s) * C_f + \max((R^* - R_s), 0) * \eta, \qquad (5.3)$$

where η is a very big number (we use 999,999 in our computer program).

For problem **P1**, we use the data in Table 5.1 as the characteristics of the available components for selection. Given $R^* = 0.92, C_f = 10, k = 5$ and $n = 3$, we have developed the GA program using Matlab GA toolbox to minimize the function given in Equation (5.3).

Table 5.1: Characteristics of available components: Database 1

ID	Utility	Reliability	Cost
1	6.0000	0.9600	16.0000
2	4.0000	0.9900	14.0000
3	5.0000	0.9300	15.0000
4	7.0000	0.9800	17.0000
5	3.0000	0.9400	13.0000
6	9.0000	0.9900	19.0000
7	5.0000	0.9700	16.0000
8	6.0000	0.9500	16.0000
9	4.0000	0.9200	15.0000
10	2.0000	0.8800	12.0000
11	7.0000	0.9500	18.0000
12	5.0000	0.8800	26.0000
13	2.0000	0.9700	29.0000
14	3.0000	0.8600	25.0000
15	4.0000	0.8500	30.0000
16	6.0000	0.8300	22.0000

When the data in Table 5.1 was used, the optimal GA solution was to select two components with ID 10 and one component with ID 5. We then changed the reliability of component 10 in Table 5.1 from 0.88 to 0.98 and ran the GA program again. This time the

optimal solution was to select three components with ID 10. The solution was reasonable, because component ID 10 was the cheapest component in the database, and the total utility of three components with ID 10 is higher than the required $k = 5$. When the reliability of component ID 10 increases to a level similar to other components, selecting component ID 10 is the best choice.

We then used the TS method to solve problem **P1** with $R^* = 0.92, C_f = 10, k = 5$, and $n = 3$. The arbitrary starting solution was to select components $x_1 = 3$, $x_2 = 10$ and $x_3 = 6$ from Table 5.1. Using component $x_1 = 3$, $x_2 = 10$ and $x_3 = 6$ to build a weighted 5-out-of-3 system, the system reliability is 0.9265, and this satisfies the system reliability requirement $R_s \geq R^*$. Starting from this initial solution, with type 1 and type 2 moves, we obtained the candidate solutions shown in Table 5.2 and Table 5.3, respectively. Selecting the solution with the minimum system cost which is 39.700 in Table 5.2, we get a new solution with $x_1 = 3$, $x_2 = 10$ and $x_3 = 10$. This is the first generation after the initial solution. Comparing it with the initial solution $x_1 = 3$, $x_2 = 10$ and $x_3 = 6$, we know that only the third component is changed. Since it has the minimum system cost, it is the best move in all the possible moves from the initial solution. The best move is then to change the third component from component 6 to component 10. At this moment, the tabu list is empty. Because the tabu list should reflect the most recent best move, the following entry is added to the tabu list:

$$39.7 \,|10\,|3$$

The first element in this entry of the tabu list is the system cost. The second element is the new component ID number which is used to substitute the old one. The third element is the

position of the substituted component. This completes the first step of implementing the TS procedure. After this, we use the present best move solution as the initial solution to go through the same process again.

Table 5.2: All possible type 1 moves from the initial solution

System cost	x_1	x_2	x_3	System cost	x_1	x_2	x_3
47.004	1	10	6	49.007	3	9	6
45.013	2	10	6	51.000	3	11	6
48.002	4	10	6	52.000	3	12	6
44.017	5	10	6	52.000	3	13	6
50.001	6	10	6	52.000	3	14	6
47.003	7	10	6	52.000	3	15	6
47.005	8	10	6	52.000	3	16	6
46.019	9	10	6	43.028	3	10	1
43.100	10	10	6	41.090	3	10	2
48.002	11	10	6	42.049	3	10	3
49.002	12	10	6	44.014	3	10	4
49.002	13	10	6	40.121	3	10	5
49.002	14	10	6	43.021	3	10	7
49.002	15	10	6	43.035	3	10	8
49.002	16	10	6	42.133	3	10	9
50.000	3	1	6	39.700	3	10	10
48.007	3	2	6	44.014	3	10	11
49.000	3	3	6	45.014	3	10	12
51.000	3	4	6	45.014	3	10	13
47.007	3	5	6	45.014	3	10	14
53.000	3	6	6	45.014	3	10	15
50.000	3	7	6	45.014	3	10	16
50.000	3	8	6				

Table 5.3: All possible type 2 moves from the initial solution

System cost	x_1	x_2	x_3
47.004	1	10	6
43.100	10	10	6
50.001	6	10	6
51.000	1	1	6
54.000	1	6	6
40.400	1	10	10
44.016	1	10	1

Using the data in Table 5.1 as the characteristics of available components, the TS method is compared with the GA method. Both methods provided the optimal result $x_1 = 5, x_2 = 10, x_3 = 10$ for problem **P1** and the minimum system cost is 37.735. In the calculations, it is observed that in order to make the result stable, at least 50 generations are needed for the GA method. The Stall Generation Limit parameter is also used to terminate the algorithm. If there is no improvement in the best fitness value after the number of generations specified by the stall generation limit, the algorithm stops. This parameter is set to be 20. The Stall Time Limit parameter is also used. If there is no improvement in the best fitness value for an interval of time in seconds specified by the stall time limit, the algorithm is also terminated. This parameter is set to be 100. In order to make the comparison fair, same values for the above parameters have been set for the TS approach. The range of each decision variable is set to be integers between 1 and 16 since there are totally 16 components in Table 5.1. The time spent on GA method is 2.1720 seconds. On the other hand, the TS method finds the optimal solution in 10 steps and uses only 0.3280 seconds.

Following the same procedure, we used the data in Table 5.1 as the characteristics of the available components, the TS method is compared with the GA method for problem **P2**, given $C^* = 54$, n=3 and k=5. In order to make the calculation easier, the Equation (5.2) and its constraints are revised to the penalty function form as follow.

Minimize:

$$-R_s = -R_s + \max(\sum_{i=1}^{n} c_i - C^*, 0) * \eta + \max(\sum_{i=1}^{n} w_i - W^*, 0) * \eta \qquad (5.4)$$

in which η is a very big number (we use 999,999 in the program).

The procedure of GA is as follows.

1. Initialization. Set the size of population and the length of the chromosome. Set $k = 0$, and generate the initial population $P(0)$.

2. Evaluation. Calculate the fitness value of each chromosome of the current population $P(k)$. Save the chromosome $B(k)$ with the best fitness value.

3. Select. Select chromosomes from the current population based on their fitness values to form a new population $P(k+1)$.

4. Cross. One point crossover is used to $P(k+1)$.

5. Mutate. Implement even mutation on $P(k+1)$.

6. Duplication. Use to replace the first chromosome in $P(k+1)$.

7. If the maximal iteration is reached, terminate the procedure and output the result. Otherwise, set $k = k+1$, and go to step (2).

For the similar reason as mentioned before, the generation step has been set to be 100 for the GA method. The stall generation limit was set to be 20 and the stall time limit was set to be 100. The initial range of the solution for GA method is still set to be [1,16]. The time spent on GA method is 2.7980 seconds. On the other hand, for the TS method, every time it still obtained the optimal result within 10 steps, although the total iteration step is set to be 100 in order to compare with GA approach fairly. Because the TS method is not sensitive to the initial solution, we still use the former one. It spent 0.4240 seconds to complete the 20 iteration steps calculation. Both methods got the optimal solution $x_1 = 4, x_2 = 6, x_3 = 4$ and the maximum system reliability is 0.98960400.

Our description of Tabu Search is much more detailed, because TS is a problem-specified approach and it is not as known in the literature as GA.

When the size of the system increases from $n=3$ to $n=10$, we also compared these two approaches. Because as the increasing of the system size, the total utility of the components and the upper limit for the total system cost should increase too, the value of k is set to be equal to $n*5$ and the C^* is set to be $n*18$ where n is the number of the component. R^* is still set to be 0.92, and C_f is equal to 10. Since in the process of solving these problems, the reliability evaluation algorithm is employed repetitively, we used the efficient UGF algorithm in these situations (n from 3 to 10 are small and medium sized systems) to calculate the system reliability. For the GA method the maximum number of generations is set to be 100, the Stall Generation Limit is set to be 50, and the Stall Time Limit is set to be 20. For the TS method, the maximum number of iterations is set to be 100. The results are listed in Table 5.4, Figure 5.4 for problem **P1** and Table 5.5, Figure 5.5 for problem **P2**. Based on the comparison results, it is clear that the TS method is more efficient and robust that the GA method.

Table 5.4: Comparison of GA and TS on CPU time for Problem **P1** based on database 1

System size (n)	Computation time (GA)	Computation time (TS)	Optimal objective function	Optimal component selection
3	2.1721	0.3283	37.7354	{5,10,10}
4	4.5710	2.0922	48.7322	{10,10,10,10}
5	9.4732	3.7850	61.2992	{10,5,10,10,10}
6	13.5452	6.2210	74.1891	{10,10,2,10,10,10}
7	17.5451	9.6340	91.6940	{5,2,5,5,5,5,10}
8	42.0582	14.3661	106.4071	{2,2,10,2,2,5,10,5}
9	61.2323	19.8123	121.6370	{10,10,4,4,10,10,5,10,2}
10	72.8324	27.2741	138.5333	{2,10,1,1,1,2,10,10,10,2}

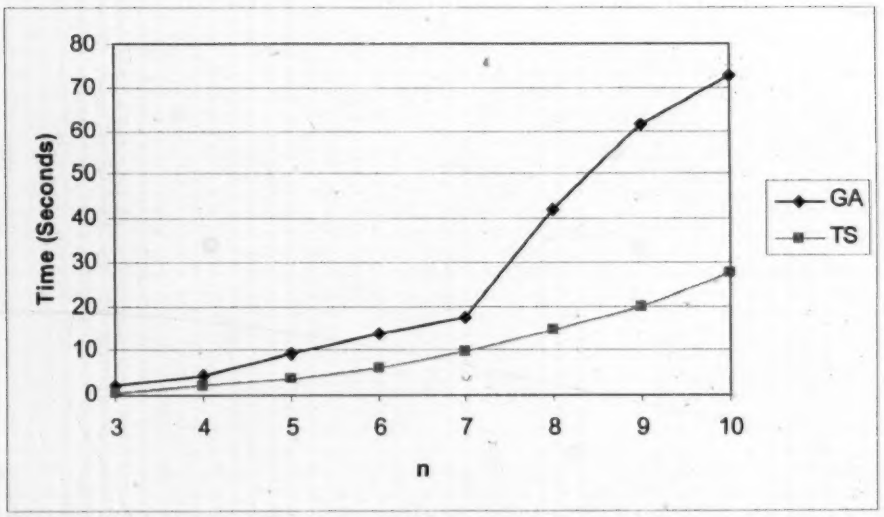


Figure 5.4: Comparison of GA and TS on CPU time for Problem **P1** based on database 1

Table 5.5: Comparison of GA and TS on CPU time for problem **P2** based on database 1

System size (n)	Computation time (GA)	Computation time (TS)	Optimal objective function	Optimal component selection
3	2.7982	0.4243	0.98960400	{4,6,4}
4	3.7031	2.0130	0.99940797	{2,6,6,6}
5	4.7650	2.7680	0.99940797	{6,6,6,10,6}
6	5.7490	4.3250	0.99957874	{1,1,6,6,4,6}
7	8.9990	6.2540	0.99993636	{2,6,6,6,4,6,4}
8	21.2652	12.4131	0.99997703	{2,6,6,6,2,6,6,6}
9	49.3112	17.9871	0.99998030	{1,6,2,6,6,6,6,1,6}
10	68.7473	26.6442	0.99999754	{2,2,6,6,4,6,6,6,6,6}

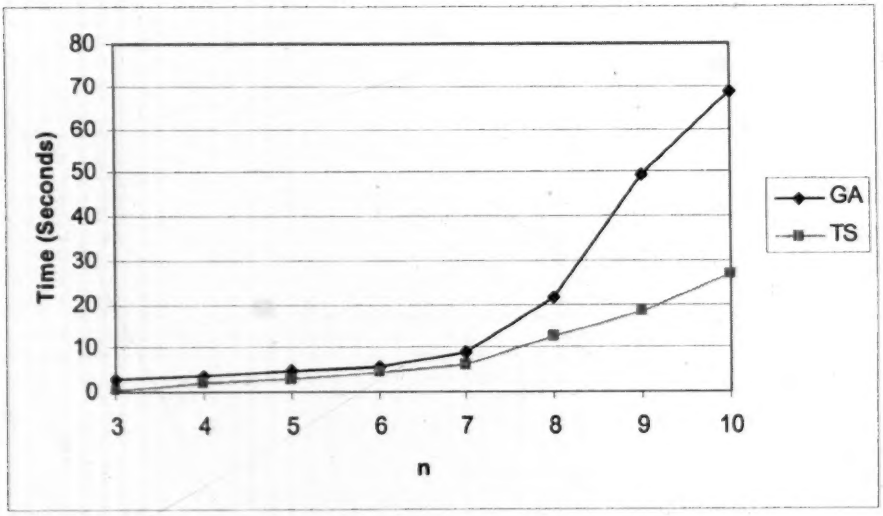


Figure 5.5: Comparison of GA and TS on CPU time for problem **P2** based on database 1

The maximum system size in the above numerical examples is 10. It is clear that the computation time for GA increases much more quickly than TS as the system size increases. Based on this trend, we have concluded that TS is more efficient than GA. Of course, if we try more numerical examples with greater system sizes, the results may be different. Either approach or both of them may not find the optimal solution. But at least based on our comparison, we can say when the system size is not very big and both GA and TS can find the optimal solution, TS is more efficient than GA.

5.4 Conclusions

We studied the optimal reliability design of the binary weighted k-out-of-n system in this chapter. Two optimal models are formulated. One is to minimize the expected total cost at the same time guarantee the system reliability bigger than a pre-specified value; the other is to maximize the system reliability with the constraints on total system cost. Genetic

Algorithms (GA) and Tabu Search (TS) methods are both used to solve the resulting optimization models. The results obtained with these two approaches for a series of example systems are compared. The results show that both are powerful tools to solve these kinds of problems, but TS is more efficient.

The binary weighted k-out-of-n system is a special case of the MS weighted k-out-of-n system, and the optimal reliability design problem studied in this chapter is the widely studied component selection optimization problem. The main contribution of this chapter is to explore the advantages of using TS relative to the most commonly used optimization tool-GA in solving this kind of problem. In the next chapter, we will move on to the more general system – MS weighted k-out-of-n system and more general optimal reliability design problem – component design problem.

References:

[1]. M. Adamantios. ReliaSoft Corporation and Tucson. Reliability Allocation and Optimization for Complex Systems. Proceedings Annual Reliability and Maintainability Symposium, Los Angeles, California, USA, January 24-27, 2000.

[2]. J. C. Chang, R. J. Chen and F.K. Hwang. A fast reliability-algorithm for the circular consecutive weighted k-out-of-n: F system. IEEE Transàctions on Reliability, 47(4): 472-474, 1998.

[3]. Y. Chen and Q. Y. Yang. Reliability of two-stage weighted-k-out-of-n systems with components in common. IEEE Transactions on Reliability, 54(3): 431-440, 2005.

[4]. D. W. Coit. System reliability optimization with k-out-of-n subsystems. International Journal of Reliability, Quality and Safety Engineering, 7(2): 129-142, 2000.

[5]. D. W. Coit. Maximization of system reliability with a choice of redundancy strategies. IIE Transactions, 35(6): 535-543, 2003.

[6]. F. Glover. Tabu Search – Part I. ORSA Journal on Computing, 1(3): 190-206, 1989.

[7]. F. Glover. Tabu Search: A Tutorial. Interfaces, 20(4): 74-94, 1990.

[8]. F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, Boston, USA, 1997.

[9]. B. Gnedenko and I. Ushakov. Probabilistic Reliability Engineering. John Wiley and Sons, Inc, NY/Chichester/Brisbane, 1995.

[10]. Y. Higashiyama. A factored reliability formula for weighted *k*-out-of-*n* system. Asia-Pacific Journal of Operational Research, 18(1): 61-66, 2001.

[11]. S. K. Konak, A. E. Smith and D. W. Coit. Efficiently solving the redundancy allocation problem using tabu search. IIE Transactions, 35: 515-526, 2003.

[12]. W. Kuo and V. R. Prasad. An annotated overview of system-reliability optimization. IEEE Transaction on Reliability, 49(2): 176-87, 2000.

[13]. G. Levitin. Optimal series-parallel topology of multi-state system with two failure modes. Reliability Engineering and System Safety, 77: 93-107, 2002.

[14]. G. Levitin (Ed.). Computational Intelligence in Reliability Engineering. New Metaheuristics, Neural and Fuzzy Techniques in Reliability, Series: Studies in Computational Intelligence, Vol. 40, Springer-Verlag, 2006.

[15]. G. Levitin and A. Lisnianski. Structure optimization of power system with bridge

topology. Electric Power Systems Research, 45: 201-208, 1998.

[16]. G. Levitin and A. Lisnianski. Reliability optimization for weighted voting system. Reliability Engineering and System Safety, 71: 131-138, 2001.

[17]. G. Levitin and A. Lisnianski. Structure Optimization of Multi-state System with Two Failure Modes. Reliability Engineering and System Safety, 72: 75-89, 2001.

[18]. A. Lisnianski, G. Levitin, and H. Ben-Haim. Structure optimization of multi-state system with time redundancy. Reliability Engineering and System Safety, 67: 103-112, 2000.

[19]. G. Levitin, A. Lisnianski and D. Elmakis. Structure optimization of power system with different redundant elements. Electric Power Systems Research, 43: 19-27, 1997.

[20]. A. Lisnianski, G. Levitin, H. Ben-Haim and D. Elmakis. Power system structure optimization subject to reliability constraints. Electric Power Systems Research, 39: 145-152, 1996.

[21]. G. Levitin, A. Lisnianski, H. Ben-Haim and D. Elmakis. Redundancy optimization for series-parallel multi-state systems. IEEE Transactions on Reliability, 47(2): 165-172, 1998.

[22]. W. Li and M.J. Zuo. Optimal design of binary weighted *k*-out-of-*n* systems. Proceedings of the 2006 Industrial Engineering Research Conference, Orlando, U.S.A, 2006.

[23]. W. Li and M.J. Zuo. Optimal design of binary weighted *k*-out-of-*n* systems. Submitted to International Journal of Reliability, Quality and Safety Engineering, 2008.

[24]. W. Li and M.J. Zuo. Reliability evaluation of multi-state weighted *k*-out-of-*n* systems. Reliability Engineering and System Safety, 93(1): 161-168, 2008.

[25]. G. D. Morley. Analysis and Design of Ring-Based Transport Network. PhD Thesis, University of Alberta, 2001.

[26]. H. Pham. Optimal design of *k*-out-of-*n* redundant systems. Microelectronics and Reliability, 32: 119-126, 1992.

[27]. H. Pham. On the optimal design of *k*-out-of-*n*: G subsystems. IEEE Transactions on Reliability, R-41(4): 572-574, 1992.

[28]. J. E. Ramirez-Marquez and D. W. Coit. A heuristic for solving the redundancy allocation problem for multi-state series-parallel. Systems Reliability Engineering and System Safety, 83(3): 341-349, 2004.

[29]. K. Reinshke and I. Ushakov. Application of Graph Theory for Reliability Analysis. Radio I Sviaz, Moscow, 1988.

[30]. K. Shen and M. Xie. On the increase of system reliability by parallel redundancy. IEEE Transactions on Reliability, 39(5): 607-611, 1990.

[31]. R. C. Suich and R. L. Patterson. *K*-out-of-*n*: G systems: some cost considerations. IEEE Transactions on Reliability, R-40(3): 259-264, 1991.

[32]. I. Ushakov. Optimal standby problem and a universal generating function. Soviet Journal Computer and system science, 25(4): 61-73, 1987.

[33]. I. Ushakov. The method of generalized generating sequences. European Journal of Operating Research, 125(2): 316-323, 2000.

[34]. J.S. Wu and R.J. Chen. An algorithm for computing the reliability of a weighted k-out-of-n system. IEEE Transactions on Reliability, R-43: 327-328, 1994.

Chapter 6

Optimal Design of Multi-State Weighted *k*-out-of-*n* Systems Based on Component Design

This chapter presents a study on design optimization of multi-state weighted k-out-of-n systems. The studied system reliability model is more general than the traditional binary k-out-of-n system model. The system and its components are capable of assuming a whole range of performance levels, varying from perfect functioning to complete failure. A utility value corresponding to each state is used to indicate the corresponding performance level. A widely studied reliability optimization problem is the "component selection problem", which involves selection of components with known reliability and cost characteristics. Less adequately addressed has been the problem of determining system cost and utility based on the relationships between component reliability, cost and utility. This chapter addresses this topic. All the optimization problems dealt within this chapter can be categorized as either minimizing the expected total system cost subject to system reliability requirements, or maximizing system reliability subject to total system cost limitation. The resulting optimization problems are too complicated to be solved by traditional optimization approaches, therefore Genetic Algorithm (GA) is used to solve them. Our results show that GA is a powerful tool for solving these kinds of problems. The materials in this chapter have been published in [16] and [18].

6.1 Introduction

6.1.1 The binary weighted k-out-of-n system model

In a binary k-out-of-n:G/F system, the system works/fails if and only if at least k components work/fail. Wu & Chen [27] generalized the binary k-out-of-n system model into the binary weighted k-out-of-n model. In a binary weighted k-out-of-n:G system, component i carries a positive integer weight, w_i, $w_i > 0$, for $i = 1,2,\ldots,n$. The total weight of all components is w, $w = \sum_{i=1}^{n} w_i$. The system works if and only if the total weight of working components is at least k, a pre-specified value. Since k is a weight, it may be larger than n because k and n are expressed using different units of measurement. Such a binary weighted k-out-of-n:G system is equivalent to a binary weighted $(w-k+1)$-out-of-n:F system wherein the system fails if and only if the total weight of failed components is at least $w-k+1$. Using these generalizations, the binary k-out-of-n system model is a special case of the binary weighted k-out-of-n system model because each component in a k-out-of-n system has a weight of one.

The "weight" used by Wu & Chen [27] refers to the contribution made by the component. A component that makes a greater contribution to the system has a greater "weight." For example, a jet plane usually has several engines. Any engine that generates a greater thrust has a higher "weight." Any engine that generates a lesser thrust has a lower "weight." Because the term "weight" has nothing to do with the component's "physical weight", in this chapter, we will avoid confusion by using the word "utility" instead of "weight".

6.1.2 The multi-state weighted k-out-of-n system models

Section 6.1.1 described the binary weighted *k*-out-of-*n*:G system model. In a binary weighted *k*-out-of-*n*:G system, the utility of the system equals the sum of the utilities of the working components. When a component has failed, its contribution to system utility is 0.

In the multi-state context, a component may be in more than two different states, varying from perfect functioning (denoted by level *M*) to complete failure (denoted by level 0). When in a different state, it may make a different contribution to the system. That is to say, every component in every possible state makes a certain contribution to the system's performance. If it has completely failed, its contribution to the system is zero.

Suppose that there are n components in a system, each of which may be in $M+1$ possible states: $\{0,1,2,\cdots,M\}$; and suppose as well that component *i*, when in state *j*, has a utility value of u_{ij}. The formal definition of Model I of the multi-state weighted *k*-out-of-*n*:G system provided by Li and Zuo [14] is given below.

Definition 6.1 [15]: The system is in state j or above if the total utility of all components is equal to or greater than k_j, a pre-specified value.

Let ϕ be the structure function of the system representing the state of the system and U_{total} the total utility of all components. Since every component in this model can contribute a utility to U_{total}, U_{total} is a function of the state of each component. Then, this definition means $\Pr\{\phi \geq j\} = \Pr\{U_{total} \geq k_j\}$. Since state 0 is the worst state of the system, we have $\Pr\{\phi \geq 0\} = 1$.

Huang et al. [3] proposed the general multi-state k-out-of-n system model without consideration of the concept of utility. According to their definition, for the system state to be no lower than a given value, j, the number of components whose state is not lower than j must be at least k_j, a pre-specified value. According to their definition, the components whose states are below j do not make any contribution to the system's being in state j or above. Building on their idea, Li and Zuo [14] defined another model of the multi-state weighted k-out-of-n:G system. According to this definition, for the system to be in state j or above, the sum of the utilities of only those components in state j or above must be no less than k_j, a pre-specified value. The difference between Model I (given in Definition 1) and Model II (to be formalized in Definition 2) is whether the components whose states are below j make any contribution to the system's being in state j or above. The formal definition of Model II is given below.

Definition 6.2 [15]: The system is in state j or above if the total utility of the components in state j or above is equal to or greater than k_j, a pre-specified value.

Let U_j be the sum of the utilities of the components whose states are j or above. Based on Definition 6.2, we then have $\Pr\{\phi \geq j\} = \Pr\{U_j \geq k_j\}$.

Applications of the multi-state weighted k-out-of-n system can be found in aircrafts, telecommunication networks, traffic systems, satellites, electric generators and their distribution systems, mining and mining distribution systems, space shuttles and computer systems. A practical example is given below. A case study can be found in [21].

Example 6.1. Conveyor systems: There are several conveyors that work together in parallel. During the life of a conveyor, its state will deteriorate from perfect to failed. When its state has deteriorated, it may not be able to carry a full load any more. That means that corresponding to different states, it has different utilities. When we require that in a system, all the conveyors must be able to transport enough coal to generate at least 100 kw no matter what state an individual conveyor may be in, we can use Model I of a multi-state weighted k-out-of-n system. When we require the whole system to carry a full load (or above a specified level), those conveyors that are unable to carry the required load may not be useful anymore. In this case, we can use Model II of a multi-state weighted k-out-of-n system.

Li and Zuo reported a recursive approach for evaluating the reliability of the two models given in Definitions 1 and 2 and also applied the Universal Generating Function (UGF) approach to solving the same problem [15]. Based on the results reported in [14], generally speaking, the recursive approach is more efficient than the UGF approach for multi-state weighted k-out-of-n system performance evaluation, however, in some special cases, UGF is more efficient than the recursive approach. For more details on reliability evaluation for these system models, please refer to [15].

In solving optimal design problems, we need to repetitively evaluate system reliability. Based on the comparison results in terms of the efficiency of the two approaches, the more efficient one will be selected for use in the following studies on optimal design. A simpler version of this chapter was published in a conference proceeding [16].

6.2 Design models

In optimal design of binary *k*-out-of-*n* systems, there are reported studies that determine the optimal system size, n [22], the optimal value of k [23], and the optimal values for n and k simultaneously [26]. The commonly used form of the objective function to be minimized is $cn + d(1 - R_s)$, where c is the cost of each component and d is the cost of system failure. This objective function is interpreted as "the expected total cost = the costs of all components + the expected cost of system failure."

In this study, we focus on the optimal design for multi-state weighted *k*-out-of-*n* systems which have not yet been studied. Two problems are addressed in this chapter. One is minimizing the expected total cost of the system subject to a minimum system reliability requirement. The other is maximizing system reliability subject to a given total budget. In studies of multi-state weighted systems, the term "reliability" has dynamic meanings, depending on the state of interest, that is the probability that the component or the system will be in state j or above. We will not treat any of the n or k_j values as decision variables; all are fixed. The decision variables are the component reliability distributions and the component utility distributions; this is because we would like to determine what components to use in the optimal system design.

According to the recursive reliability evaluation approach and the UGF approach described earlier, the reliability of the multi-state weighted *k*-out-of-*n* system is a function of the component utility distribution matrix and the component reliability matrix: $R_s = R(\boldsymbol{U}, \boldsymbol{P})$, where the component reliability distribution matrix, $\boldsymbol{P}$, is:

$$\begin{bmatrix} p_{10} & p_{11} & p_{12} & \cdots & p_{1M} \\ p_{20} & p_{21} & p_{22} & \cdots & p_{2M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n0} & p_{n1} & p_{n2} & \cdots & p_{nM} \end{bmatrix},$$

where p_{ij} is the probability of component i being in state j, $i \in \{1,2,\cdots,n\}$ and $j \in \{0,1,2,\cdots,M\}$, and the utility distribution matrix, $\boldsymbol{U}$, is:

$$\begin{bmatrix} u_{10} & u_{11} & u_{12} & \cdots & u_{1M} \\ u_{20} & u_{21} & u_{22} & \cdots & u_{2M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{n0} & u_{n1} & u_{n2} & \cdots & u_{nM} \end{bmatrix},$$

where u_{ij} is the utility of component i in state j, $i \in \{1,2,\cdots,n\}$ and $j \in \{0,1,2,\cdots,M\}$. Thus, the decision variables are $\boldsymbol{U}$ and $\boldsymbol{P}$.

Furthermore, there are two types of problem in the field of optimal reliability design.

(1) There are discrete component choices with known characteristics (cost, utility, reliability). The objective is to determine which components to use.

(2) Component utilities and component reliabilities are all treated as design variables. System cost is a pre-defined function of component utilities and component reliabilities. This is the more general situation.

The first type of problem is the component selection problem that has been well studied in recent years in system optimal design for various structures. Shen & Xie [25] studied the component selection problem to maximize the reliability of a parallel system. Coit [1] studied the problem of redundancy allocation using discrete component choices for multi-state series-parallel systems. The objective function was to maximize system reliability

subject to constraints on system cost and system physical weight. Ramirez-Marquez & Coit [24] continued the study of the redundancy allocation problem using discrete component choices for multi-state series-parallel systems. The objective function was to minimize system cost subject to system reliability requirements. Li and Zuo [15] studied the problem of component selection for binary weighted k-out-of-n systems. In a series of research work by Levitin [7], Levitin & Lisnianski [9], [10], [11], Levitin et al. [12], Levitin et al. [13] and Lisnianski et al. [12], the following component selection problems were studied:

(1) maximizing system reliability without considering costs at all,

(2) minimizing system cost subject to requirements on system reliability.

The second type of problem represents the more general situation where components can be designed and produced to desired specifications. If the manufacturers not only assembled the system but also designed components themselves, they would be interested in solving the second type of problem. Though manufacturers may not specifically design multi-state components such as 3-state components or 4-state components, these components may turn out to be able to assume more than two possible states. In the binary context, engineers try their best to design components that never fail, that is, components with only a single (perfect) state. In reality that is impossible. We need to study failure modes because any component may fail. In the multi-state context, we need to evaluate the design of a component to fully understand that the component could experience different states with different probabilities. How do we change the design so that the probabilities of a component being in any given state can be optimized? These are the issues that motivated us to pursue this study. In addition, reliability of any product

must be defined and evaluated by the customer. The purpose of our research is to develop the reliability models which can capture the experience and requirements of the customer with the product. If customers need a single specific system that is not available on the market, we believe manufacturers will develop the multi-state component on which it is based, even though designing and manufacturing a component is considered less cost effective than buying it on the market. This is because any work done to meet the needs of customers attracts more customers and, thus may generate profits for the manufacturers. If done intelligently this could more than repay the investment in design and manufacture. So far, however, almost all reported studies have focused on only the first type of problem. The second type of problem is much more complicated. The reason is that it is hard to develop a function that describes the relationship between component cost, reliability and utility. Generally, the empirical data are not adequate to fitting such a function. Even when they are, different systems' data may provide different functions.

This chapter will address the second type of problem. We will first extend the relationship between component reliability and component cost from the binary system to the multi-state weighted k-out-of-n system, and then, develop the relationship between component utility and component cost based on the analysis of empirical data. Next, we will combine these two relationship functions to express component cost as a function of component reliability and component utility. Finally, we will use the obtained function to conduct optimal design of multi-state weighted k-out-of-n systems. To the best of the authors' knowledge, this is the first time the relationship between component utility and cost has been developed based on the analysis of empirical data, and this is the first reported attempt to systematically address the second type of reliability design problems.

The optimization models formulated here are for optimal design of the multi-state weighted k-out-of-n system which offers the advantage of being a much more general system than the traditional binary k-out-of-n system. Both system reliability and system total cost are considered in the models. Note that, the cost considered here is the expected total cost of the system, not just the cost of the components.

For the second type of problem, we need to develop a function to describe the relationship between the component cost, component reliability and component utility. Mettas et al. [20] gave the component cost as a function of component reliability for a binary component as follows:

$$c = \exp[(1-f)\cdot\frac{p-p_{min}}{p_{max}-p}] \tag{6.1}$$

where

- p_{min} : minimum reliability of the component
- p_{max} : maximum reliability of the component
- f : feasibility of increasing the reliability of the component
- c : cost of the component
- p : reliability of the component.

The cost here is the cost of acquiring the component, not the cost or consequences caused by its failure. The feasibility parameter, f, is a positive constant, which represents the difficulty in increasing the component's reliability. It assumes values between 0 and 1 [20]. Several methods can be used to obtain a feasibility value. Weighting factors for allocating reliability have been proposed by many authors and can be used to quantify feasibility [4]. These weights depend on certain factors of influence [4] such as the

complexity of the component, the state of the art, the operational profile, the criticality, etc. Engineering judgment based on past experience, supplier quality, supplier availability, etc., can also be used in determining values of these parameters [20]. When this equation is extended to the ith component of the multi-state weighted k-out-of-n system Model I, p should be replaced by $\sum_{j=1}^{M} p_{i,j}$, and Equation (6.1) should be revised to read:

$$cI_i^I = \exp[(1-f_i^I)\cdot\frac{\sum_{j=1}^{M} p_{i,j} - P_{i_{min}}^I}{P_{i_{max}}^I - \sum_{j=1}^{M} p_{i,j}}] \tag{6.2}$$

where

- $P_{i_{min}}^I$: minimum probability of component i being in a state higher than state 0
- $P_{i_{max}}^I$: maximum probability of components i being in a state higher than state 0
- f_i^I : feasibility of increasing the probability of component i in Model I being in a state higher than state 0
- cI_i^I : cost of component i in Model I, in terms of considering the relationship between component reliability and component cost.

There may be other ways of extending Equation (6.1) to the multi-state case, but the form we proposed is as close to Equation (6.1) as one can be. The proposed Equation (6.2) actually relates the cost of buying a given component to the total probability that the component is not in the completely failed state (which is state 0). This basically sums all the probabilities that the component is in states 1, 2, …, through M. This total probability is then treated in Equation (6.2) in the same way "reliability" is treated in Equation (6.1).

If we let $M = 2$, Equation (6.2) reduces to Equation (6.1), demonstrating that Equation (6.1) is a special case of Equation (6.2).

We also need to obtain the relationship between the cost of a component and its utility. In practice, an empirical relationship is often not available or the empirical data are not adequate to fitting such a function. We did not try to fit the function of cost to reliability and utility based on empirical data. Instead, we tried to observe some trends in the practical data and used mathematical equations to describe these trends. Table 6.1 and Table 6.2 give the prices for different capacities of Western Digital Hard Disks (Ultra ATA-100, 7200-RPM, 8MB, 3.5 IDE, HDD) and Archos MP3 Player Compact Flash Memory Cards as recorded on Feb. 15, 2006. The hard disk data were obtained from Western Digital Corporation's website [27], and the memory card data were obtained from 18004memory's website [29]. Although the data are so limited that we can not fit a usable function, we can still observe some trends in these data:

(1) The price increases as the product capacity (utility) increases.

(2) The more the product capacity (utility) increases, the faster the price increases.

Table 6.1: The price for the Western Digital Hard Disks (Ultra ATA-100 7200-RPM 8MB 3.5 IDE HDD) (www.wdc.com)

Capacity (GB)	40	80	120	160	200	250	320
Price (US dollars)	69	79	99	109	129	159	219

Table 6.2: The price for the Archos MP3 player Compact Flash Memory Card (www.18004memory.com)

Capacity (GB)	0.128	0.256	0.512	1	2	4
Price (US dollars)	16	25	35	59	119	299

Based on the two examples above, it can be seen that the cost of a component may increase exponentially in relation to its capacity or utility. Thus the exponential function

can be used to approximately describe the relationship between the component's cost and its utility as follows.

$$c = g \cdot \exp(u - u_{min}) \tag{6.3}$$

where:

- u_{min} : minimum utility of the component
- g : feasibility of increasing the utility of the component
- c : cost of the component
- u : utility of the component.

Equation (6.3) follows the exponential form used in Equation (6.1). Actually, we are not promoting the use of any specific function form; we are promoting the idea of relating the cost of a component to its utility. It is the concept shown in Equation (6.3) that is of the utmost importance. One can use the linear function form, the polynomial function form, etc, to represent such relationships if they are applicable. The feasibility parameter, g , is a positive constant that represents the difficulty of increasing a component's utility. The more difficult it is to improve the utility of the component, the greater the cost. In addition, it makes no sense to have a 0 utility product, therefore usually the value of u is equal to or greater than a given value, u_{min}.

The proposed utility function given in Equation (6.3) satisfies the following requirements:

(1) Component cost is a monotonically increasing function of component utility.

(2) The derivative of component cost (with respect to utility) is a monotonically increasing function of component utility.

This means that the function given in Equation (6.3) matches our earlier observations of the practical data. When this function is extended to the i^{th} component of the multi-state weighted *k*-out-of-*n* system Model I, the utility of the binary component should be replaced by the "expected utility" of the multi-state component which is the weighted average of the component utilities in different states $\sum_{j=0}^{M} p_{i,j} * u_{i,j}$. Thus Equation (6.3) should be revised to read:

$$c2_i^I = g_i^I \cdot \exp(\sum_{j=0}^{M} p_{i,j} * u_{i,j} - u_{i_{min}}^I) \tag{6.4}$$

where:

- $U_{i_{min}}^I$: minimum value for the weighted average of the component i's utilities in all possible states
- g_i^I : feasibility of increasing the utility of component i in Model I
- $c2_i^I$: cost of the component i in Model I in terms of the relationship between component utility and component cost.

It can be seen that Equation (6.2) ignores the effect of component *utility* on the component cost. Equation (6.4) also ignores the effect of component *state probability* on component cost. Each considers only one factor and relates the trend of the component cost to the changing of that factor. Clearly, a more reasonable approach is to consider the utility and the state probability simultaneously. The following function is proposed to describe this relationship. Actually this function adds Equation (6.2) and Equation (6.4) together.

$$c_i^I = g_i^I \cdot \exp(\sum_{j=0}^{M} p_{i,j} * u_{i,j} - U_{i_{min}}^I) + \exp[(1 - f_i^I) \cdot \frac{\sum_{j=1}^{M} p_{i,j} - P_{i_{min}}^I}{P_{i_{max}}^I - \sum_{j=1}^{M} p_{i,j}}] \quad (6.5)$$

Given $U_i{}^I = \sum_{j=0}^{M} p_{i,j} * u_{i,j}$ and $P_i^I = \sum_{j=1}^{M} p_{i,j}$, Equation (6.5) can be rewritten as:

$$c_i^I = g_i^I \cdot \exp(U_i{}^I - U_{i_{min}}^I) + \exp[(1 - f_i^I) \cdot \frac{P_i^I - P_{i_{min}}^I}{P_{i_{max}}^I - P_i^I}] \quad (6.6).$$

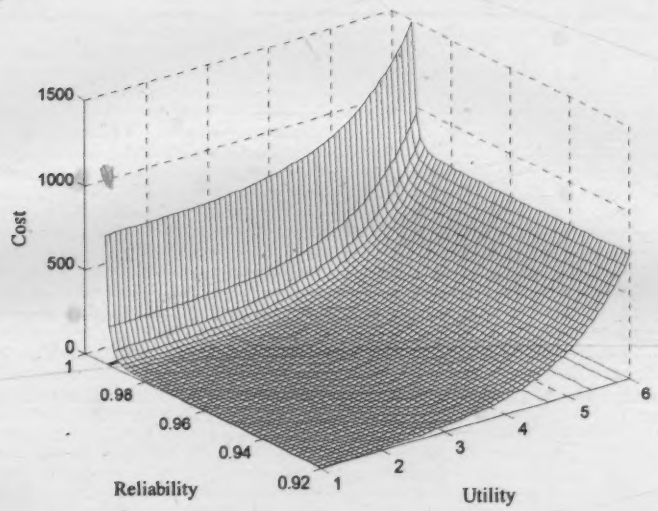


Figure 6.1: Component cost as a function of reliability and utility

We are not claiming that the proposed functions in Equation (6.2) to (6.6) hold in the majority of cases. They are generated based on the exponential trends in the two examples cited, partly because this form is also used in Equation (6.1) by Mettas et al [20]. The basic idea for the component cost function is to formulate the relationship between component cost, reliability and utility. The proposed functions outline the importance of developing such relationships, and the proposed forms are examples of such relationships.

The parameters used in the cost function are explained as follows. The feasibility parameters (feasibility of increasing component reliability, f_i^I, and feasibility of increasing component utility, g_i^I) are constants that represent the difficulty in increasing a component's reliability and utility relative to other components. Depending on design complexity, technological limitations, etc., some components can be very hard to improve, relative to other components. Clearly, the more difficult it is to improve the reliability and utility of a component, the greater the cost will be. Examining the effect of feasibility on cost function in Equation (6.2), it can be seen that the lower the feasibility value, f_i^I, the more rapidly the cost function approaches infinity. Examining the effect of feasibility on cost function in Equation (6.4), it can be seen that the greater the feasibility value, g_i^I, the more rapidly the cost function approaches infinity. The maximum component reliability, $P_{i_{max}}^I$; minimum component reliability, $P_{i_{min}}^I$; and minimum component weighted average utility, $U_{i_{min}}^I$, act as scale parameters for the cost function. In accordance with customer requirements, component reliability and weighted average utility should be at least equal to or above a minimum level. Thus we have defined the minimum component reliability and minimum component weighted average utility. The maximum component reliability is a parameter that shows that technically the probability value for a component's reliability cannot reach 1. The weighting factor method (Section 15.7 in [4]) and engineering judgment based on past experience, supplier quality, supplier availability, etc., can be used in determining values of these parameters [20].

Given $g_i^I = 5$, $f_i^I = 0.4$, $U_{i_{min}}^I = 1$, $P_{i_{min}}^I = 0.93$, $P_{i_{max}}^I = 0.9999$, $U_i^I \in [1,6]$ and $P_i^I \in [0.93, 0.9999]$, the function given in Equation (6.6) is plotted in Figure 6.1. In this figure, reliability refers to P_i^I and utility refers to U_i^I.

When we consider Model II given in Definition 2, the component cost function is as follows:

$$c_i^{II_j} = g_i^{II_j} \cdot \exp(U_i^{II_j} - U_{i_{min}}^{II_j}) + \exp[(1 - f_i^{II_j}) \cdot \frac{P_i^{II_j} - P_{i_{min}}^{II_j}}{P_{i_{max}}^{II_j} - P_i^{II_j}}] \quad (6.7)$$

where

- $U_i^{II_j} = \sum_{r=j}^{M} p_{i,r} * u_{i,r}$ and $P_i^{II_j} = \sum_{r=j}^{M} p_{i,r}$
- $P_{i_{min}}^{II_j}$: minimum probability of component i being in state j or above
- $P_{i_{max}}^{II_j}$: maximum probability of component i being in state j or above
- $f_i^{II_j}$: feasibility of increasing the probability of component i being in state j or above
- $U_{i_{min}}^{II_j}$: minimum utility of the weighted average of component i's utilities in all states
- $g_i^{II_j}$: feasibility of increasing the utility of component i for Model II
- $c_i^{II_j}$: cost of component i for Model II if the component can contribute to the system in state j or above

As we have mentioned already, the difference between Models I and II is whether the components whose states are below j are making any contribution for the system to be

in state j or above. In Model II, only the components whose states are in state j or above can make contributions for the system to be in state j or above. It is the reason why we have $U_i^{II_j} = \sum_{r=j}^{M} p_{i,r} * u_{i,r}$ and $P_i^{II_j} = \sum_{r=j}^{M} p_{i,r}$. The meanings of parameters in Equation (6.7) are the same as in Equation (6.6).

The component cost functions we developed describe a general behavior of the component cost as a function of component reliability and utility, to be used in cases where an actual function is not available. If there is an actual function for a specific system, of course, the actual function should be used. Unfortunately, there seldom is one. In addition, the cost functions provided above are the first ones to describe the general relationship between component cost, reliability and utility. As already mentioned, this does not mean our cost functions are the only possible ones. Even in the binary reliability context, as mentioned in [20], that there are other forms of cost function besides Equation (6.1). Other general forms were suggested by [4] and [6]. Other researchers may someday suggest other general forms for the multi-state weighted system. They may use different parameters and forms, but again, the basic idea for these functions will be the same – to describe the general behavior of cost as a function of reliability and utility.

With the relationship given in Equations (6.6) and (6.7), the design optimization problems can be formulated as given below.

Problem P1:

Minimize:

$$C_s^{T_j} = \sum_{i=1}^{n} c_i^X + (1 - R_j^T(k_j, n)) \cdot C_j^T \qquad (6.8)$$

Subject to:

$$R_j^T(k_j,n) \geq \hat{R}_j^T$$

$$\sum_{j=0}^{M} p_{i,j} = 1, 0 \leq p_{i,j} \leq 1 (i = 1,2,\cdots,n; j = 0,1,2,\cdots,M)$$

$$u_{i,0} = 0, u_{i,j} \geq 0 (i = 1,2,\cdots,n; j = 1,\cdots,M)$$

Problem P2:

Maximize:

$$R_j^T(k_j,n) \tag{6.9}$$

Subject to:

$$C_s^{T_j} = \sum_{i=1}^{n} c_i^X + (1 - R_j^T(k_j,n)) \cdot C_j^T \leq \hat{C}_s^{T_j}$$

$$\sum_{j=0}^{M} p_{i,j} = 1, 0 \leq p_{i,j} \leq 1 (i = 1,2,\cdots,n; j = 0,1,2,\cdots,M)$$

$$u_{i,0} = 0, u_{i,j} \geq 0 (i = 1,2,\cdots,n; j = 1,\cdots,M)$$

The notation in the above models is:

- $C_s^{T_j}$: the expected total cost when the system is in state j or above
- $R_j^T(k_j,n)$: the probability of the system being in state j or above, given k_j, j and n
- C_j^T: the cost of the system state being below state j
- $\hat{R}_j^T$: the minimum required probability for the system to attain a state of j or above
- $\hat{C}_s^{T_j}$: the upper limit for the total cost when the system attains a state of j or above

- $T \in \{I, II\}, X \in \{I, II_j\}$. Specifically, when $T = I$, X must be equal to I. Under these conditions, the above two models (P1 and P2) are for a multi-state weighted k-out-of-n system Model I and the component cost function is given by Equation (6.6). When $T = II$, X must be equal to II_j. Under these conditions, the above two models are for the multi-state weighted k-out-of-n system Model II and the component cost function is given by Equation (6.7).

In the above models, $P_{i_{min}}^X$, $P_{i_{max}}^X$, $U_{i_{min}}^X$, g_i^X, f_i^X, k_j, j, $\hat{C}_s^{T_j}$, $\hat{R}_j^T$, C_j^T, M and n are all parameters. The decision variables are $\boldsymbol{U}$ and $\boldsymbol{P}$.

In the models, we assume that the component utility distribution can be fully controlled. In practice, it is hard to say that the component utility distribution can be controlled in all situations. With the rapid development of today's manufacturing technology, we do, however, have much better control over the utility distribution and state distribution than ever before. With this rapid development of technology, we need advanced models that provide new directions and new challenges for technological development. Furthermore, the models brought forward in this chapter are the foundation for further research. For this reason, we suppose that the component utility can be fully controlled. In the future, it will be easy to put constraints on component utility in order to describe situations in which it can only be partially controlled.

Applications of these optimization models can be found widely in industries, including but not limited to oil and gas, transportation, and telecommunication. Example 6.1, which has been described in Section 6.1.2 will be used to illustrate the possible application of these models. If designers want to reduce cost and at the same time keep reliability at a

given level, because a conveyor system is a multi-state weighted k-out-of-n system as described in Section 6.1.2, the proposed optimization model Problem P1 is suitable for this case. In practice, the choice between Model I and II depends on the requirements of the conveyor system. For instance, if all the conveyors are required to carry a full load, Problem P1 for Model II should be used; if there is no specific requirement for the state of the conveyors, Problem P1 for Model I should be used. Similarly, if there is a fixed budget for building a conveyor system, and the aim is to obtain the greatest system reliability possible under that budget, Problem P2 is suitable for the application.

6.3 Solution approach and an illustration example

Reported optimization approaches for solving reliability-based design problems for the binary systems include dynamic programming, integer programming, mixed integer programming, non-linear programming, heuristics, and metaheurisctcs. A literature survey on these approaches was provided by [5].

The family of metaheuristic optimization techniques includes Simulated Annealing, Tabu Search, Genetic Algorithm (GA), Evolutionary Strategies, Ant Colony, Immune Algorithm, and Swarm Optimization. The applications of some of these optimization techniques can be found in [7]. These methods have been found to be more powerful for solving large-scale and complex optimization problems. They perform better than traditional optimization algorithms in finding global optimal solutions. Especially, GA is the most widely used metaheuristic approach to solving large-scale complex optimization problems.

GA has two key advantages:

Flexibility in modeling the problem. GA has no strict mathematical requirements, such as the derivative requirement for objective functions and constraints. The only requirement is that the objective functions and constraints can be evaluated in some way. GA is also suitable for dealing with those problems that include discrete design variables.

Global optimization ability. GA has been recognized as one of the most effective approaches to searching for the global optimal solution.

In this chapter, we use the GA approach to solve the design problems formulated in Section 6.2. In order to make the calculation easier, the objective functions and constraints in problems **P1** to **P2** have been written into the penalty function forms as follows.

Problem P1:

Minimize:

$$\sum_{i=1}^{n} c_i^T + (1 - R_j^T(k_j, n)) \cdot C_j^T + \max(\hat{R}_j^T - R_j^T(k_j, n), 0) * \eta \qquad (6.10)$$

Problem P2:

Minimize:

$$-R_j^T(k_j, n) + \max(\sum_{i=1}^{n} c_i^T + (1 - R_j^T(k_j, n)) \cdot C_j^T - \hat{C}_s^{T_j}, 0) * \eta \qquad (6.11)$$

In these objective functions, η is a very large number. (We use 999999 in the programs.) These objective functions are all still subjected to:

$$\sum_{j=0}^{M} p_{i,j} = 1, 0 \le p_{i,j} \le 1 (i = 1, 2, \cdots, n; j = 0, 1, 2, \cdots, M)$$

$$u_{i,0} = 0, u_{i,j} \ge 0 (i = 1, 2, \cdots, n; j = 1, \cdots, M).$$

A system with two components ($n=2$) wherein each component has four states ($M=3$) is used as an illustration of the optimization problems and their solutions.

Given $j=2$, $k_2=3$, $R_2^T=0.9$, $C_2^T=10$ and $\hat{C}_s^{T_2}=9$, the other parameters used in this example are those given in Table 6.3.

Table 6.3: Parameters used in the example

$g_1^I=g_2^I$	$f_1^I=f_2^I$	$U_{1_{min}}^{I_j}=U_{2_{min}}^{I_j}$	$P_{1_{min}}^I=P_{2_{min}}^I$	$P_{1_{max}}^I=P_{2_{max}}^I$
1	0.99	1	0.93	0.99999
$g_1^{II_j}=g_2^{II_j}$	$f_1^{II_j}=f_2^{II_j}$	$U_{1_{min}}^{II_j}=U_{2_{min}}^{II_j}$	$P_{1_{min}}^{II_j}=P_{2_{min}}^{II_j}$	$P_{1_{max}}^{II_j}=P_{2_{max}}^{II_j}$
1	0.99	1	0.93	0.99999

After developing the objective functions and their constraints, we used the GA toolbox [2] in Matlab R2006a to solve the problems. We ran these programs using a computer with 2.00GHZ AMD Opteron I Processor and 3.00 GB RAM under the Windows XP operating system. We set the population data type at double and population size at 20. Elite count and crossover fraction were all set at the default values in Matlab R2006a GA toolbox. The uniform function was used as the population creation function. The adaptive feasible function was used as the mutation function. The rank function was used as the scaling function. The scattered function was used as the crossover function. The stochastic uniform function was used as the selection function. The stopping criteria were 1000 generations, 500 stall generations, 100 stall time limit, and 10^{-9} function tolerance. The reason we set the stopping criteria this way is that we found the results were convergent for all the systems we tried (n=2, 10 and 20). For more details about GA toolbox in Matlab R2006a and for the terminology used here, please refer to [2] and the

Matlab R2006a Help Document. The optimization results for Models I and II are listed in Table 6.4 and Table 6.5, respectively.

There are 14 decision variables in each of the above optimization problems, all of which are non-linear non-integer optimization problems. It is difficult to use traditional optimization approaches to solve these problems, however, by using GA, they were all solved in about 7 seconds. In this example, the value of n is 2. When the value of n increases to 10 and all the other parameters remain the same, there are 70 variables in each optimization model, and completing the computation takes about 4 minutes. When the value of n increases to 20 and all the other parameters remain the same, there are 140 variables in each model, and completing the computation takes about half an hour. Clearly, even though the system is large, the GA approach can still solve these optimization problems.

Table 6.4: Optimization results in Model I

	Problem P1					**Problem P2**			
Reliability	0.9781					0.9927			
Cost	5.9753					8.8525			
U	0	2.0000	2.0000	2.0001		0	1.1185	3.0000	3.0000
	0	1.0000	1.0000	1.0026		0	1.9994	1.9996	2.0000
P	0.0127	0.3291	0.3291	0.3291		0.0056	0.3326	0.3314	0.3314
	0.0087	0.3305	0.3299	0.3303		0.0107	0.3302	0.3299	0.3302

Table 6.5: Optimization results in Model II

	Problem P1					**Problem P2**			
Reliability	0.9690					0.9770			
Cost	5.9473					8.0905			
U	0	0.0000	2.0423	2.0456		0	0.0000	1.9425	3.0000
	0	0.0000	0.9604	0.9616		0	0.0000	1.2508	1.7610
P	0.0162	0.0095	0.4844	0.4908		0.0012	0.0006	0.5003	0.4986
	0.0039	0.0034	0.4968	0.4968		0.0027	0.0431	0.4907	0.4645

An interesting observation was that the systems for Problem P1 in Table 6.4 and Table 6.5 actually both have only two states; combining states with the same utilities makes sense, because a binary component (system) is more cost-efficient than a multi-state component (system), assuming that both meet the reliability requirement. From our results we can see that the optimization problems that minimize the total cost have a lower total cost than optimization problems that maximize the probability of the system being at or above a given state. On the other hand, the optimization problems that maximize the probability that a system state will be at or above a given state are better able to attain that goal than optimization problems that minimize the total cost. This is reasonable, obviously when the only objective is to minimize the total cost, we sacrifice some requirements for the probability of the system being at or above a given state by just giving a constraint for that probability. That constraint is usually selected not to be too high in order to avoid infeasibility. Similarly, when the only objective is to maximize the probability of the system being at or above a given state, some requirements for limiting the total cost will be sacrificed, resulting in a larger total cost allowance. After investigating the optimization results, the decision-maker may decide that the total cost goal can be relaxed a little more in order to improve system reliability. Since we are dealing with a multi-state weighted system, the term system reliability means the probability of the system being at or above a given state.

6.4 Conclusions

In this chapter, we have defined two optimization problems for the multi-state weighted k-out-of-n system which includes two models – Model I and Model II. All the optimization problems studied in this chapter can be categorized as either minimizing the

expected total system cost subject to system reliability requirements, or maximizing system reliability subject to total cost limitations. Genetic Algorithm (GA) was used to solve these problems. The results showed that GA was a powerful tool for solving these kinds of problems.

In our optimization models, there exist mutually conflicting goals: system reliability and total system cost. We treat one goal as the objective function of the optimization model and the other as the constraint. It is very difficult to specify in advance the value that should be given to the goal as the constraint. After a solution is obtained, we often need to modify the constraint value to find a better trade-off between goals such as system reliability and system cost. Finding the most appropriate constraint value is a trial and error process that can be time-consuming. In the next chapter, we will use a multi-objective optimization tool, physical programming [19], to optimize different goals simultaneously.

References:

[1] D.W. Coit. Maximization of system reliability with a choice of redundancy strategies. IIE Transactions, 35(6): 535-543, 2003.

[2] C. Houck, J. Joines and M. Kay. A genetic algorithm for function optimization: a Matlab implementation. North Carolina State University – Industrial Engineering Technical Report 95-09, 1995.

[3] J. Huang, M.J. Zuo and Y. Wu. Generalized multi-state *k*-out-of-*n*: G systems. IEEE Transactions on Reliability, 49(1): 105 – 111, 2000.

[4] D. Kececioglu. Reliability Engineering Handbook, Volume 2, PTR Prentice Hall, 1991.

[5] W. Kuo and V.R. Prasad. An annotated overview of system-reliability optimization. IEEE Transactions on Reliability, 49(2): 176-87, 2000.

[6] L.M. Leemis. Reliability, Probabilistic Models and Statistical Methods, Prentice-Hall, 1995.

[7] G. Levitin. Optimal series-parallel topology of multi-state system with two failure modes. Reliability Engineering and System Safety, 77: 93-107, 2002.

[8] G. Levitin (Ed.). Computational Intelligence in Reliability Engineering. New Metaheuristics, Neural and Fuzzy Techniques in Reliability, Series: Studies in Computational Intelligence, Vol. 40, Springer-Verlag, 2006.

[9] G. Levitin and A. Lisnianski. Structure optimization of power system with bridge topology. Electric Power Systems Research, 45: 201-208, 1998.

[10] G. Levitin and A. Lisnianski. Reliability optimization for weighted voting system. Reliability Engineering and System Safety, 71: 131-138, 2001.

[11] G. Levitin and A. Lisnianski. Structure optimization of multi-state system with two failure modes. Reliability Engineering and System Safety, 72: 75-89, 2001.

[12] A. Lisnianski, G. Levitin and H. Ben-Haim. Structure optimization of multi-state system with time redundancy. Reliability Engineering and System Safety, 67: 103-112, 2000.

[13] G. Levitin, A. Lisnianski and D. Elmakis. Structure optimization of power system with different redundant elements. Electric Power Systems Research, 43: 19-27, 1997.

[14] G. Levitin, A. Lisnianski, H. Ben-Haim and D. Elmakis. Redundancy optimization for series-parallel multi-state systems. IEEE Transactions on Reliability, 47(2): 165-172, 1998.

[15] W. Li and M.J. Zuo. Optimal design of binary weighted *k*-out-of-*n* systems, Proceedings of the 2006 Industrial Engineering Research Conference, Orlando, Florida, U.S, 2006.

[16] W. Li and M.J. Zuo. Optimal design of multi-state weighted *k*-out-of-*n* systems based on component Design. Proceedings of the European Safety and Reliability Conference, Stavanger (Norway), 25-27 June, 2007

[17] W. Li and M.J. Zuo. Reliability evaluation of multi-state weighted *k*-out-of-*n* systems. Reliability Engineering and System Safety, 93(1): 161-168, 2008.

[18] W. Li and M.J. Zuo. Optimal design of multi-state weighted *k*-out-of-*n* systems based on component design. Reliability Engineering & System Safety, Accepted for Publication (available online), 2008.

[19] A. Messac. Physical programming: effective optimization for computational design. AIAA Journal, 34(1): 149-158, 1996.

[20] A. Mettas. ReliaSoft Corporation and Tucson, Reliability allocation and optimization for complex systems. Proceedings of the Annual Reliability and Maintainability Symposium, Los Angeles, California, USA, January 24-27, 2000.

[21] B. Natvig, S. Sormo, A.T. Holen and G. Hogasen. Multi-state reliability theory – a case-study. Advances in Applied Probability, 18 (4): 921-932, 1986.

[22] H. Pham. Optimal design of *k*-out-of-*n* redundant systems. Microelectronics and Reliability, 32: 119-126, 1992.

[23] H. Pham. On the optimal design of *k*-out-of-*n*: G subsystems. IEEE Transactions on Reliability, 41(4): 572-574, 1992.

[24] J.E. Ramirez-Marquez and D.W. Coit. A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems. Systems Reliability Engineering and System Safety, 83(3): 341-349, 2004.

[25] K. Shen and M. Xie. On the increase of system reliability by parallel redundancy. IEEE Transactions on Reliability, 39(5): 607-611, 1990.

[26] R.C. Suich and R.L. Patterson. *K*-out-of-*n*: G systems: some cost considerations. IEEE Transactions on Reliability, 40(3): 259-264, 1991.

[27] J.S. Wu and R.J. Chen. An algorithm for computing the reliability of a weighted *k*-out-of-*n* system. IEEE Transactions on Reliability, 43: 327-328, 1994.

[28] Western Digital Corporation's website, www.wdc.com.

[29] 18004memory's website, www.18004memory.com.

Chapter 7

Optimal Design of Multi-state Weighted Series-Parallel System Using Physical Programming and Genetic Algorithms

In this chapter, we report a study of the reliability optimal design for multi-state weighted series-parallel systems. Such a system and its components are capable of assuming a whole range of levels of performance, varying from perfect functioning to complete failure, and there is a component utility corresponding to each component state. This system model is more general than the traditional binary series-parallel system model. Its logical structure is different from multi-state weighted k-out-of-n systems as studied in Chapter 4 and Chapter 6. Its definition has already been provided in Section 3.2.1 of Chapter 3. More details are given later in this chapter. The so-called component selection reliability optimal design problem which involves selection of components with known reliability characteristics and cost characteristics has been widely studied. However, the problem of determining system cost and system utility based on the relationships between component reliability and cost and component utility and cost has not been adequately addressed. We call it component design reliability optimal design problem which has been studied in Chapter 6 for the MS weighted k-out-of-n system and continued the research in this chapter for the multi-state weighted series-parallel systems. Furthermore, comparing to the traditional single-objective optimization model, the optimization model we proposed in this chapter is a multi-objective optimization model which is used to maximize expected system performance utility and system reliability while minimizing

system cost simultaneously. GA is used to solve the proposed physical programming based optimization model. An example is used to illustrate the flexibility and effectiveness of the proposed approach over the single-objective optimization method. A simpler version of materials in this chapter has been published in [15]. An extended version of materials in this chapter has been submitted to IEEE Transaction on Reliability [17].

7.1 Introduction

The traditional reliability theories focus on binary reliability models, allowing for only two possible states for a system and its components: perfect functionality and complete failure. However, many of the real-world systems are composed of multi-state (MS) components, which have different performance levels and several failure modes. Such systems are called multi-state systems (MSS). In a MSS reliability model, the system and each component may experience states in the set $\{0,1,2,...,M\}$. When a multi-state component or system is in a certain state j, it can produce a certain amount of benefit. Such benefit is called the component or system utility. When the utility of the component and system is considered in the MSS model, the model is a multi-state weighted system model.

In practical situations involving reliability optimization, there often exist mutually conflicting goals such as maximizing system utility and minimizing system cost and weight [11]. Most reported multi-state optimization models treat one goal as the objective function and the other goals as constraints. However, it is very difficult to specify in advance the constraint values for the goals used as constraints. After a solution is

obtained, we often need to modify these constraint values to find a better trade-off between different goals. However, finding the most appropriate constraint values is a trial and error process and there is no clear guideline as to how to converge to the right set of constraint values. Particularly, when there are many constraint values that need to be specified, it is almost impossible to find the most appropriate values for these constraints.

Physical programming has proved its effectiveness in addressing a wide array of multi-objective optimization problems ([24]; [25]; [26]; [25] and [30]). Tian, Zuo and Huang [33] and Tian and Zuo [34] studied the redundancy allocation problem for multi-state series-parallel systems using physical programming. The problem Tian, Zuo and Huang [33] and Tian and Zuo [34] studied is to optimize the structure of the multi-state series-parallel systems. The problem we will study here is to optimize the parameters of the components in which the multi-state weighted series-parallel system structure has already been fixed.

There are two classes of problems in the reliability optimal design field:

1. There are discrete component choices with know characteristics (cost, utility, reliability). The objective is to determine which components to use.

2. The more general situation: component utilities and component reliabilities are all treated as design variables. System cost is a pre-defined function of component utilities and component reliabilities.

The first class is a component selection problem which has been studied in recent years in optimal design of systems of various structures. Shen & Xie [32] studied the component selection problem for binary parallel systems to maximize system reliability. Specifically,

they gave the answers for two questions: one is what is the increase of system reliability after parallel redundancy of a specific component, the other is which component is to be chosen for parallel redundancy so that the increase of system reliability is largest. Liu, Zuo and Meng [20] formulated the optimization model for continuous multi-state series-parallel systems to determine the number of redundancies in order to maximize the system's expected utility function subject to constraints on system cost. The design goal is also achieved by discrete choices made from components available in the market. In a series of research work by Lisnianski and Levitin, most of which are recorded in their book [19], they studied maximizing system reliability without considering costs at all and minimizing system cost subject to requirement on system reliability. The system structures they studied include multi-state series-parallel systems, multi-state weighted voting system and multi-state linear consecutively-connected systems. Coit [7] studied the redundancy allocation problem with discrete component choices for binary series-parallel systems. The objective function was to maximize system reliability subject to constraints on system cost and system physical weight. Massim et al. [22] proposed how to use the ant colony algorithm to choose an optimal multi-state series–parallel power structure configuration in order to minimize total investment cost subject to availability constraints. They defined the availability as the probability that the multi-state series–parallel system will be in the state with capacity level greater than or equal to a specific demand at the specified moment. Li and Zuo [13] studied the component selection optimal design problem for binary weighted k-out-of-n systems. Nourelfath and Ait-Kadi [29] studied the minimal cost configuration of a multi-state series-parallel system subject to a specified maintenance policy. A special multi-state series-parallel system is

constructed by the binary components which can make the system exhibit multi-state behavior. Ramirez-Marquez and Coit [31] studied the redundancy allocation problem of this kind of system with discrete component choices from a list of products available in the market. The objective was to minimize system cost subject to system reliability requirement. Gupta and Agarwal [9] studied the optimization problem for this kind of system to maximize system reliability subject to system cost. They used GA together with the penalty technique to solve the formulated problem. Agarwal and Gupta [1] continued their research to present the ant colony algorithm for a homogeneous series-parallel system, exhibiting a multi-state behavior, to minimize the cost in order to provide a desired level of reliability. Meziane et al. [23] described how to implement the ant colony algorithm for finding the optimal multi-state series–parallel power system configurations in order to maximize system reliability subject to system performance and cost constraints. Furthermore, Massim et al. [21] used the ant colony method to solve the multi-stage expansion problem for multi-state series-parallel systems. The objective is to minimize the whole investment costs over the study period while satisfying availability or performance constraints by adding additional components selected from a list of available products.

The second class of problems represents the more general situation where we can also design and produce the components to our desired specifications. If the manufacturers not only assemble the system but also produce components by themselves, they would be interested in solving the second class of problems. However, based on our study, all reported studies focused on the first class of problems. The second class is much more complicated than the first. The reason is that it is hard to develop a function to describe

the relationship between component cost, reliability and utility. Generally, the empirical data are not adequate to fit such a function. Even they are, different systems' data may provide different functions. Li and Zuo [14] and [18] studied the second class of problems for multi-state weighted k-out-of-n systems by formulating the component cost function through analyzing some practical data of the Western Digital Hard Disks and the Archos MP3 player Compact Flash Memory Card.

In this chapter, we focus on the second class of problems for multi-state weighted series-parallel systems. We present a multi-objective optimization model, which optimizes the goals of expected system utility, investment cost and system reliability simultaneously. The decision variables are the component reliability distribution matrix P and the component utility distribution matrix U as we would like to determine the parameters of the components to be used in the optimal system design.

7.2 Multi-state weighted series-parallel system

Barlow and Wu [3] defined the coherent multi-state series-parallel system model. A MS series-parallel system consists of subsystems, S_1, S_2, ..., S_N, connected in series. Each subsystem, S_i, has n_i different MS weighted components connected in parallel. So there are totally n_s components in the system where $n_s = \sum_{i=1}^{N} n_i$. Each component may be in $M+1$ states: $\{0,1,2,\cdots,M\}$. Component i has a probability p_{ij} in state j. If the coherent multi-state system is a series system, then the state of its "worst" component is assigned to the system as the system state. If the coherent multi-state system is a parallel system, then the state of its "best" component is assigned to the system as the system

state. The "worst" component here means the component with the lowest state. The "best" component here means the component with the highest state.

When this coherent MS series-parallel system is extended to the MS weighted series-parallel system. The system is still a coherent system and consists of subsystems, S_1, S_2, ..., S_N, connected in series. Each subsystem, S_i, has n_i different MS weighted components connected in parallel. Each component may be in $M+1$ states: $\{0,1,2,\cdots,M\}$. Component i has a probability p_{ij} in state j, and there is a utility value u_{ij} corresponding to each component state. The utility value u_{ij} is used to describe the component performance capability in that state. The component reliability distribution matrix P is:

$$\begin{bmatrix} p_{10} & p_{11} & p_{12} & \cdots & p_{1M} \\ p_{20} & p_{21} & p_{22} & \cdots & p_{2M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n_s 0} & p_{n_s 1} & p_{n_s 2} & \cdots & p_{n_s M} \end{bmatrix},$$

where p_{ij} is the probability of component i in state j, $i \in \{1,2\cdots,n_s\}$, and $j \in \{0,1,2\cdots,M\}$.

and the component utility distribution matrix U is:

$$\begin{bmatrix} u_{10} & u_{11} & u_{12} & \cdots & u_{1M} \\ u_{20} & u_{21} & u_{22} & \cdots & u_{2M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{n_s 0} & u_{n_s 1} & u_{n_s 2} & \cdots & u_{n_s M} \end{bmatrix}.$$

where u_{ij} is the utility of component i in state j, $i \in \{1,2\cdots,n_s\}$ and $j \in \{0,1,2\cdots,M\}$.

The structure of a multi-state weighted series-parallel system is as shown in Figure 7.1.

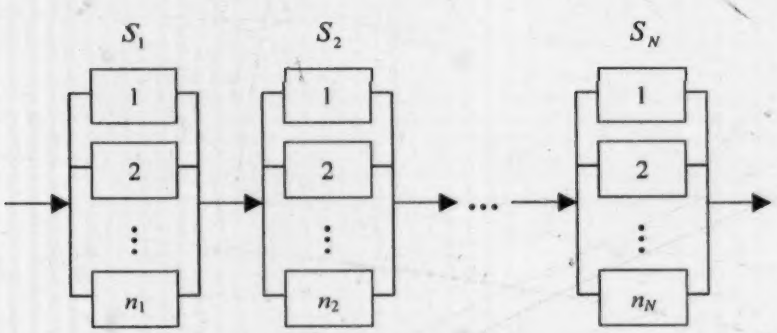


Figure 7.1: Structure of a multi-state weighted series-parallel system

In the traditional MS series-parallel system model, because there is no parameter to describe the component (system) performance capacity, we have to assume that the people using the model know the component (system) performance capacity corresponding to each state. When the MS series-parallel system model is extended to the MS weighted series-parallel system model, the above assumptions can be removed which makes the model more effective and flexible. The published reliability optimal design research work for MS series-parallel systems has been summarized in Section 1. This chapter on the reliability optimal design of MS weighted series-parallel systems makes the following contributions. It is the first reported work on the component design problem for MS weighted series-parallel systems which have stronger modeling ability than the traditional MS series-parallel systems. The system reliability, total investment cost and expected system utility are considered together in one optimization model. Furthermore, the expected system utility is based on the component utility distribution, while Aven [2], Tian, Zuo and Huang [33] and Tian and Zuo [34], assumed it to be a given value.

Li and Zuo [14], [14] and [18] studied the reliability evaluation and component design optimization of MS weighted k-out-of-n systems separately. The component selection

reliability design of the binary weighted k-out-of-n system which is a special case of MS weighted k-out-of-n system was addressed in Li and Zuo [13]. The MS weighted series-parallel system has a different logical structure from the MS weighted k-out-of-n systems. The definition of MS weighted series-parallel systems and the reliability evaluation approach are described in the following subsections.

7.2.1 The definitions of MS weighted series-parallel systems

MS weighted series systems

Suppose that the MS weighted components are connected in series. The system is in state j or above if and only if every component's utility is equal to or bigger than the given value k_j, a pre-specified value, where $k_j \le k_{j+1}$, for $j = 0,1,2,\cdots,M-1$.

Based on this definition, the MS weighted series system utility is the utility of the worst component. For example, two three-state components are connected in series. Their state distributions are given in Table 7.1 and Table 7.2.

Table 7.1: State distribution of component 1

State	0	1	2
Utility	0	3	4
Probability	0.1	0.2	0.7

Table 7.2: State distribution of component 2

State	0	1	2
Utility	0	5	7
Probability	0.3	0.5	0.2

Given $k_0 = 0$, $k_1 = 3$ and $k_2 = 4$, the system utility distribution is listed in Table 7.3.

Table 7.3: Utility distribution of the MS weighted series system

System state	Component utility		System utility	Probability	
State 0	0	0	0	0.03	0.37
	0	5		0.05	
	0	7		0.02	
	3	0		0.06	
	4	0		0.21	
State 1	3	5	3	0.10	0.14
	3	7		0.04	
State 2	4	5	4	0.35	0.49
	4	7		0.14	

The system has a probability of 0.37 in state 0 with a utility of 0, a probability of 0.14 in state 1 with a utility of 3, and a probability of 0.49 in state 2 with a utility of 4.

In order to illustrate the application of this model, an example is given as follows.

Example 7.1: There are three cities: A, B and C. There is a fiber optic cable connecting city A and city B, and another one connecting city B and city C. Each cable may work under three states: failed, marginal and perfect. The bandwidth of each cable is its utility, since it is the index describing the performance capability of a cable. When any cable fails, the bandwidth is 0 Mbps (megabit per second). When the cable between A and B is working in the marginal state, the bandwidth is 5 Mbps. When it is working in the perfect state, the bandwidth is 10 Mbps. When the cable between B and C is working in the marginal state, the bandwidth is 10 Mbps. When it is working in the perfect state, the bandwidth is 20 Mbps. Considering each cable as a multi-state weighted component, the communication system from city A to city C via city B is a multi-state weighted series system. The utility of the system is the smaller value of the two cables' utilities.

MS weighted parallel systems

Suppose that the MS weighted components are connected in parallel. The parallel system is in state j or above if at least one component's utility is equal to or bigger than a given value k_j, a pre-specified value, where $k_j \le k_{j+1}$, for $j = 0,1,2,\cdots,M-1$.

Based on this definition, the MS weighted parallel system utility is the maximum of all the component utilities. Given $k_0 = 0$, $k_1 = 3$ and $k_2 = 5$, still using the two three-state components in Table 7.1 and Table 7.2, the MS weighted parallel system utility distribution is listed in Table 7.4.

Table 7.4: Utility distribution of the MS weighted parallel system

System state	Component utility		System utility	Probability	
State 0	0	0	0	0.03	0.03
State 1	3	0	3	0.06	0.27
	4	0	4	0.21	
State 2	4	5	5	0.35	0.70
	0	5		0.05	
	3	5		0.10	
	4	7	7	0.14	
	0	7		0.02	
	3	7		0.04	

From Table 7.4, we can see when the system is in state 1, the system utility can be 3 and 4. When the system is in state 2, the system utility can be 5 or 7. Therefore, weighted average utility values may be calculated to describe the system's average performance ability in state 1 and 2. The weighted average system utility in state 1 can be calculated as follow.

$$(3*0.06+4*0.21)/0.27=3.7778$$

The weighted average system utility in state 2 can be calculated as follow.

$$[5*(0.35+0.05+0.1)+7*(0.14+0.02+0.04)]/0.70= 5.5714$$

Therefore, this MS weighted parallel system has a probability of 0.70 in state 2 with a weighted average utility of 5.5714. It also has a probability of 0.27 in state 1 with a weighted average utility of 3.7778, and has a probability of 0.03 in state 0 with a utility of 0.

In order to illustrate the application of this model, an example is given as follow.

Example 7.2: There are two cities: A and B. Two fiber optic cables connect them together. Each cable can work under three possible states: failed, marginal and perfect. The bandwidth of each cable is its utility. When a cable fails, the bandwidth is 0. When the first cable between A and B is working in the marginal state, the bandwidth is 5 Mbps. When it is working in the perfect state, the bandwidth is 10 Mbps. When the second cable between A and B is working in the marginal state, the bandwidth is 10 Mbps. When it is working in the perfect state, the bandwidth is 20 Mbps. Considering each cable as a multi-state weighted component, the two cable communication system between city A and B is a multi-state weighted parallel system. The utility of the system is the utility of the better cable of the two.

MS weighted series-parallel systems

Combining the MS weighted parallel system and the MS weighted series system, we get the MS weighted series-parallel system. Such a system has N subsystems connected in series. Subsystem i has n_i components connected in parallel, $1 \le i \le N$. A MS weighted series-parallel system is in state j or above if in every parallel subsystem at least one component's utility is equal to or bigger than a given value k_j, which also means every subsystem has to be in state j or above.

According to the definitions of the MS weighted series system and the MS weighted parallel system, the utility of a MS weighted parallel system is defined to be the utility of the best component in the system, and the utility of a MS weighted series system is defined to be the utility of the worst component in the system. Hence, the system utility of the MS weighted series-parallel system is:

$$\gamma(\boldsymbol{u}) = \min_{1 \le i \le N} \max_{1 \le j \le n_i} u_{ij} \tag{7.1}$$

where u_{ij} is the utility of the jth component in the ith subsystem, and $\boldsymbol{u}$ is the vector representing the utilities of all components in the MS weighted series-parallel system. The structure of an example MS weighted series-parallel system is illustrated in Figure 7.2. The state distributions of component 1 and 2 are already listed in Table 7.1 and Table 7.2. The state distributions of component 3 and 4 are listed in Table 7.5 and Table 7.6. We will use this example to illustrate how to calculate the MS weighted series-parallel system utility.

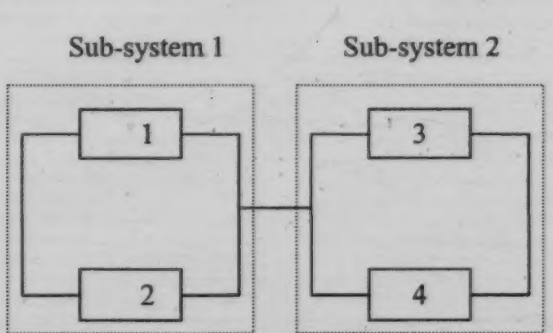


Figure 7.2: An example of MS weighted series-parallel system

Table 7.5: State distribution of component 3

State	0	1	2
Utility	0	3	5
Probability	0.1	0.3	0.6

Table 7.6: State distribution of component 4

State	0	1	2
Utility	0	1	4
Probability	0.3	0.3	0.4

Given $k_0 = 0$, $k_1 = 3$ and $k_2 = 5$, the example MS weighted series-parallel system utility distributions in each state are listed separately in Table 7.7 to Table 7.9.

Table 7.7: Utility distribution of the MS weighted series-parallel system in state 0

Component utility	System utility	Probability		Component utility	System utility	Probability	
0 0 0 0	0	0.0009	0.0591	0 7 0 1	1	0.0006	0.0291
0 0 0 1		0.0009		3 0 0 1		0.0018	
0 0 0 4		0.0012		3 5 0 1		0.0030	
0 0 3 0		0.0027		3 7 0 1		0.0012	
0 0 3 1		0.0027		4 0 0 1		0.0063	
0 0 3 4		0.0036		4 5 0 1		0.0105	
0 0 5 0		0.0054		4 7 0 1		0.0042	
0 0 5 1		0.0054		0 5 0 1		0.0015	
0 0 5 4		0.0072					
0 5 0 0		0.0015					
3 0 0 0		0.0018					
0 7 0 0		0.0006					
3 5 0 0		0.0030					
3 7 0 0		0.0012					
4 0 0 0		0.0063					
4 5 0 0		0.0105					
4 7 0 0		0.0042					

From Table 7.7, we can see that the MS weighted series-parallel system has a probability of 0.0882=0.0591+0.0291 in state 0. In state 0, the system has a probability of 0.0591 with a utility 0 and has a probability of 0.0291 with a utility 1. So the weighted average system utility is 0.3299= (0.0591*0+0.0291*1)/ 0.0882.

Table 7.8: Utility distribution of the MS weighted series-parallel system in state 1

Component utility	System utility	Probability		Component utility	System utility	Probability	
0 5 3 0		0.0045		0 5 0 4		0.0020	
0 5 3 1		0.0045		0 5 3 4		0.0060	
0 7 3 0		0.0018		0 7 0 4		0.0008	
0 7 3 1		0.0018		0 7 5 4		0.0048	
3 0 0 4		0.0024		3 5 0 4		0.0040	
3 0 3 0		0.0054		3 5 3 4		0.0120	
3 0 3 1		0.0054		4 7 0 4		0.0056	
3 0 3 4		0.0072		4 7 3 4		0.0168	
3 0 5 0		0.0108		0 7 3 4	4	0.0024	0.2764
3 0 5 1		0.0108		3 7 0 4		0.0016	
3 0 5 4	3	0.0144	0.2202	3 7 3 4		0.0048	
3 5 3 0		0.0090		4 0 0 4		0.0084	
3 5 3 1		0.0090		4 0 3 4		0.0252	
3 7 3 0		0.0036		4 0 5 0		0.0378	
3 7 3 1		0.0036		4 0 5 1		0.0378	
4 0 3 0		0.0189		4 0 5 4		0.0504	
4 0 3 1		0.0189		4 5 0 4		0.0140	
4 5 3 0		0.0315		4 5 3 4		0.0420	
4 5 3 1		0.0315					
4 7 3 0		0.0126					
4 7 3 1		0.0126					

Table 7.9: Utility distribution of the MS weighted series-parallel system in state 2

Component utility	System utility	Probability	
0 5 5 0		0.0090	
0 5 5 1		0.0090	
0 5 5 4		0.0120	
0 7 5 0		0.0036	
0 7 5 1		0.0036	
3 5 5 0		0.0180	
3 5 5 1		0.0180	
3 5 5 4		0.0240	
3 7 5 0	5	0.0072	0.4152
3 7 5 1		0.0072	
3 7 5 4		0.0096	
4 5 5 0		0.0630	
4 5 5 1		0.0630	
4 5 5 4		0.0840	
4 7 5 0		0.0252	
4 7 5 1		0.0252	
4 7 5 4		0.0336	

From Table 7.8 we can see that the MS series-parallel system has a probability of 0.4966=0.2202+0.2764 in state 1. In state 1, the system has a probability of 0.2202 with a utility 3 and has a probability of 0.2764 with a utility 4. Thus the weighted average system utility is 3.5566= (0.2202*3+0.2764*4)/0.4966.

From Table 7.9 we can see the system has a probability of 0.4152 to be in state 2. The system utility is 5.

The state distribution of this example MS series-parallel system is summarized in Table 7.10.

Table 7.10: State distribution of the MS series-parallel system

State	0	1	2
Average Utility	0.3299	3.5566	5
Probability	0.0882	0.4966	0.4152

Here we have used the enumeration method to calculate the average system utility. In our future research, we will try to find more efficient methods.

In order to illustrate the application of this model, two examples are given as follow.

Example 7.3: Extending the system in Example 7.2 to three cities: A, B and C. The two fiber optic cables connecting B and C are the same as the two cables between A and B. So totally there are four cables in the system. Considering each cable as a multi-state weighted component, the four cable communication system between city A and C is a multi-state weighted series-parallel system. The utility of each MS weighted parallel subsystem (cables between A and B and cables between B and C) is the utility of the better cable in that subsystem. And the utility of this MS weighted series-parallel system is the utility of the worse of the two subsystems.

Example 7.4: Several sets of conveyors work together in series because of the long distance to transmit coal. At the same time, in every set, there are several conveyors work together in parallel. During the life of a conveyor, its state will deteriorate from perfect working to failed. When its state deteriorates, it can not work at full capacity any more. That means corresponding to different states, it has different utilities. This system can also be modeled as a MS weighted series-parallel system.

7.2.2 Reliability evaluation of MS weighted series-parallel systems

According to the definition of the MS weighted series-parallel system, the probability that the system in state j or above ($j = 0, 1, ..., M$) is

$$\Pr\left(\phi(\boldsymbol{u}) \geq j\right) = \Pr\left(\gamma(\boldsymbol{u}) \geq k_j\right) = \prod_{i=1}^{N} \Pr\left(\gamma_i(\boldsymbol{u}_i) \geq k_j\right) \tag{7.2}$$

where $\boldsymbol{u}$ is the vector representing the utilities of all components in the system, $\phi(\boldsymbol{u})$ is the structure function of the system, $\phi(\boldsymbol{u}) = 0, 1, 2, \cdots, M$, $\gamma(\boldsymbol{u})$ is the system utility function, $\boldsymbol{u}_i$ denotes the utilities of all the components in the ith parallel subsystem, $\gamma_i(\boldsymbol{u}_i)$ is the ith parallel subsystem's utility function, and k_j is the pre-specified utility value. Equation (7.2) says that for the system to be in state j or above, the system utility should not be less than the pre-specified value k_j. Thus, the utility of each parallel subsystem should not be less than k_j. As illustrated in Equation (7.2), the reliability of the multi-state weighted series-parallel system is a function of component utility distribution matrix U and component reliability distribution matrix P: $\Pr(\phi(x) \geq j) = R_s^j(U, P)$.

As shown in Equation (7.2), in order to calculate the reliability of the whole system, we have to calculate the reliability of each parallel subsystem first. Given $\phi_i(x_i)$ is the structure function of the ith parallel subsystem, the probability of this subsystem being in state j or above is:

$$\begin{aligned}&\Pr\left(\phi_i(x_i) \geq j\right) = \Pr\left(\gamma_i(u_i) \geq k_j\right) \\ &= \Pr\left\{\left(u_{i1} \geq k_j\right) \cup \left(u_{i2} \geq k_j\right) \cdots \cup \left(u_{ir} \geq k_j\right) \cup \cdots \cup \left(u_{in_i} \geq k_j\right)\right\}\end{aligned} \tag{7.3}$$

where n_i is the number of components in the ith parallel subsystem and u_{ir} is the utility of the rth component in the ith parallel subsystem. Here we suppose that all components are independent.

According to the above analysis, the MS weighted series-parallel system can be decomposed into a binary series-parallel system. Actually, since the MS weighted series-parallel system is extended from the coherent MS series-parallel system, it is also a coherent system. As the coherent MS series-parallel system can be decomposed into a binary series-parallel system [3], [4] and [5], the MS weighted series-parallel system can also be decomposed into a binary series-parallel system. Thus the reliability evaluation method of the binary series-parallel system can be used to evaluate the probability distribution of the MS weighted series-parallel system.

We can use the following method to decompose a MS weighted parallel system into a binary parallel system: for all the component states with a utility value not less than k_j, adding the state probability together to be the reliability of that component in the corresponding binary parallel system. By this way, we can form a binary parallel system.

The reliability of this binary parallel system is equal to $Pr\{\phi_i(x_i) \geq j\}$ - the probability of the corresponding MS weighted series-parallel system to be in state j or above. With the same method, given different k_j ($j = 0, 1, ..., M$), the probability of the system to be in state j ($j = 0, 1, ..., M$) or above can be calculated.

After we get $Pr\{\phi_i(x_i) \geq j\}$ for $i=1, 2, \ldots N$ for every MS weighted parallel subsystem, we can use Equation (7.2) to calculate the probability of the MS weighted series-parallel system in state j or above.

7.3 Design model

For the second class of optimal design problems, we need to have a function to describe the relationship between the component cost, component reliability and component utility. The following function developed in chapter 6 can be used to describe this relationship.

$$c_i = g_i \cdot \exp(U_i - U_{i_{min}}) + \exp[(1 - f_i) \cdot \frac{P_i - P_{i_{min}}}{P_{i_{max}} - P_i}] \qquad (7.4)$$

where $U_i = \sum_{j=0}^{M} p_{i,j} \cdot u_{i,j}$ and $P_i = \sum_{j=1}^{M} p_{i,j}$,

- $p_{i_{min}}$: minimum probability of component i being in states higher than state 0
- $p_{i_{max}}$: maximum probability of components i being in states higher than state 0
- f_i : feasibility of increasing the probability of component i being in states higher than state 0 $(0 \leq f_i \leq 1)$
- $U^1_{i_{min}}$: minimum value for the weighted average utility of component i in all possible states

- g_i : feasibility of increasing the weighted average utility of component i in all possible states $(g_i \geq 0)$
- c_i : cost of component i
- P_i : the probability of component i being in states higher than state 0

Actually Equation (7.4) is just Equation (6.5) without superscript I. The parameters and the physical meaning of the cost function have been explained in chapter 6.

With the relationship given above, the total investment cost can be calculated as.

$$C_s = \sum_{i=1}^{n_s} c_i, \tag{7.5}$$

where $n_s = \sum_{i=1}^{N} n_i$, c_i is the cost of component i which is a function of component utility and reliability distributions.

The following index of system utility is used to measure the performance of a multi-state weighted series-parallel system:

$$U_s = \sum_{j=0}^{M} U_j \cdot \Pr(\phi(\boldsymbol{x}) = j), \tag{7.6}$$

where U_s is the expected system utility, and U_j is the weighted average utility when system is in state j. The way to calculate U_j has been addressed in Section 2.1. Although the form of Equation (7.6) is similar to the expected system utility function in Aven [2], they are different. Aven [2] assumed U_j is a given value in their system utility function. However, in our system utility function, U_j is calculated based on the component utility distribution. This is an advantage of our model. Aven [2] assumed the

people who use that model such as the engineers already knew the system performance capacity corresponding to each state. It may not always be true. In his model, U_j value can only be obtained based on engineering experiences which may not be accurate enough. In our model, U_j value is calculated based on the component utility distribution. Aven's assumption has been relaxed here to make the model more flexible.

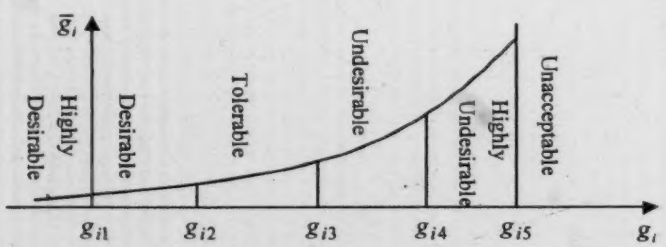
The system reliability, total investment cost and expected system utility are three important parameters in the system optimal design. Based on Equations (7.2), (7.5) and (7.6), given a specific value of j and k_j, the system reliability, total investment cost and expected system utility are all functions of component utility distribution matrix U and component reliability matrix P. The MS weighted series-parallel system reliability here means the probability that the system is at state j or above, given a specific value of j and k_j. The multi-objective optimization problem can then be described as how to find the optimal values of U and P to get a balance among the following three different objectives, minimizing system cost, maximizing system performance utility, and maximizing system reliability simultaneously. In the following, the physical programming approach is used to address the conflicting nature of these different objectives. Genetic algorithm is to be used to solve the physical programming based optimization models.

Physical programming (Messac, [24]) provides the means for direct expression of the designer's preference, which fundamentally impacts the design process. Rather than spending substantial efforts tweaking weights and re-optimizing until a given set of preferences is achieved, the designer is allowed to concentrate more on the physical

problem at hand and less on the art of converging to satisfactory weights. Within the physical programming procedure, design metrics which are characteristics or properties of a system or process, being designed, are classified into three classes:

1) Class-1: Smaller-Is-Better (SIB),
2) Class-2: Larger-Is-Better (LIB), and
3) Class-3: Center-Is-Better (CIB).

Physical Programming is explicitly incorporates the designer's preferences on each design metric into the optimization process. For each class, there are two so-called class functions, one soft and one hard, with respect to each class. The hard class functions are used to represent the constraints, while the soft class functions become additive constituent components of the aggregate objective function (to be minimized) of the optimization model. Consider for example the case of Class-1 soft class function (Class 1-S), the qualitative meaning of the preference function is depicted in Figure 3. The value of the design metric, g_i, is on the horizontal axis, and the corresponding class function, $\overline{g}_i$, is on the vertical axis. A lower value of the preference function is better than a higher value thereof.



Class 1-S

Figure 7.3: Qualitative meaning of soft class function

Physical programming allows the DM to express ranges of differing levels of preference with respect to each design metric with more flexibility and specificity than by simply declaring minimize, maximize or equal to. For Class 1-S, shown in Figure 7.3, the ranges are defined as follows.

Highly desirable range: $g_i \le g_{i1}$,

Desirable range: $g_{i1} \le g_i \le g_{i2}$,

Tolerable range: $g_{i2} \le g_i \le g_{i3}$,

Undesirable range: $g_{i3} \le g_i \le g_{i4}$,

Highly undesirable range: $g_{i4} \le g_i \le g_{i5}$,

Unacceptable range: $g_i \ge g_{i5}$.

The parameters g_{i1} through g_{i5} are physically meaningful constants associated with each design metric i. What the DM needs to do in the physical programming framework is just to specify the values of the parameters g_{i1}, g_{i2}, g_{i3}, g_{i4}, and g_{i5} for each design metric i, and the class function can be completely determined by these parameters. For further details of physical programming, please refer to Messac [24].

Two key advantages of physical programming are: (1) Once the designer's preferences are articulated, obtaining the corresponding optimal design is a non-iterative process – in stark contrast to conventional weight-based methods. (2) It provides the means to reliably employ optimization with minimal prior knowledge thereof.

In our proposed physical programming based optimization models, the design metrics under consideration are system utility, system cost and system reliability. The system

utility and system reliability, which are to be maximized, are both described using the class-2S class function in the physical programming approach framework. The system cost, which is to be minimized, is described using the class-1S class function.

The multi-objective optimization model of the multi-state weighted series-parallel system is formulated as:

$$
\begin{aligned}
&\min_{U,P} g(U,P) = \log_{10}\left\{\frac{1}{3}\left[\overline{g}_U\left(U_s(U,P)\right)+\overline{g}_R\left(R^j{}_s(U,P)\right)+\overline{g}_C\left(C_s(U,P)\right)\right]\right\} \\
&s.t. \\
&U_s(U,P) \geq U_0 \\
&R^j{}_s(U,P) \geq R_0 \\
&C_s(U,P) \geq C_0 \\
&\sum_{j=0}^{M} p_{ij} = 1, 0 \leq p_{ij} \leq 1 \ \left(i = 1,2,\cdots,n_s;\ j = 0,1,2,\cdots,M\right) \\
&u_{i0} = 0, u_{ij} \geq 0 \ \left(i = 1,2,\cdots,n_s;\ j = 1,2,\cdots,M\right)
\end{aligned}
\tag{7.7}
$$

where $g(U,P)$ is the aggregate objective function, $\overline{g}_U$, $\overline{g}_R$ and $\overline{g}_C$ are the class functions of system utility, system reliability and system cost, respectively, U is the component utility distribution matrix, and P is the component reliability matrix. U_0, R_0 and C_0 are the constraint values, which are equal to the boundaries of the acceptable ranges of the corresponding design metrics.

In order to illustrate the advantages of the multi-objective model, single-objective models are also built as follow to compare with the multi-objective optimal model. The single-objective model which the system reliability is used as the objective, and investment cost and expected system utility are used as constraints is as follow:

$$
\begin{aligned}
&\max_{U,P} R_s^j(U,P) \\
&s.t. \\
&U_s(U,P) \geq U_0 \\
&C_s(U,P) \leq C_0 \\
&\sum_{j=1}^{M} p_{ij} = 1,\ 0 \leq p_{ij} \leq 1\ (i = 1,2,\cdots,n_s;\ j = 0,1,2,\cdots,M) \\
&u_{i0} = 0,\ u_{ij} \geq 0 (i = 1,2,\cdots,n_s;\ j = 1,2,\cdots,M)
\end{aligned}
\tag{7.8}
$$

The single-objective model which the expected system utility is used as the objective, and investment cost and system reliability are used as constraints is as follow:

$$
\begin{aligned}
&\max_{U,P} U_s(U,P) \\
&s.t. \\
&R_s^j(U,P) \geq R_0 \\
&C_s(U,P) \leq C_0 \\
&\sum_{j=1}^{M} p_{ij} = 1,\ 0 \leq p_{ij} \leq 1\ (i = 1,2,\cdots,n_s;\ j = 0,1,2,\cdots,M) \\
&u_{i0} = 0,\ u_{ij} \geq 0\ (i = 1,2,\cdots,n_s;\ j = 1,2,\cdots,M)
\end{aligned}
\tag{7.9}
$$

The single-objective model which the investment cost is used as the objective, and system reliability and expected system utility are used as constraints is as follow:

$$
\begin{aligned}
&\min_{U,P} C_s(U,P) \\
&s.t. \\
&U_s(U,P) \geq U_0 \\
&R_s^j(U,P) \geq R_0 \\
&\sum_{j=1}^{M} p_{ij} = 1,\ 0 \leq p_{ij} \leq 1\ (i = 1,2,\cdots,n_s;\ j = 0,1,2,\cdots,M) \\
&u_{i0} = 0,\ u_{ij} \geq 0 (i = 1,2,\cdots,n_s;\ j = 1,2,\cdots,M)
\end{aligned}
\tag{7.10}
$$

7.4 Solution approach and an illustration example

Genetic Algorithm (GA) is the most widely used meta-heuristic approach to solve large complex optimization problems and it has been proved to perform much better than traditional optimization algorithms in finding global optimal solutions.

GA has two key advantages: (1) Flexibility in modeling the problem. GA has no strict mathematical requirements, such as derivative requirement, on the objective functions and constraints. The only requirement is that the objective functions and constraints can be evaluated in some way. GA is also suitable for dealing with those problems including discrete design variables. (2) Global optimization ability. GA has been recognized as one of the most effective approaches in searching for the global optimal solution.

In this chapter, we use the GA approach to solve the design problems formulated in Section 3. A system with four components ($n_s = 4$) wherein each component has four states ($M = 3$) is used as an example to illustrate the optimization problems and their solutions. The system structure is as shown in Figure 7.2.

Given, k_0=0, k_1=1, k_2=2, k_3=3 and $j = 2$, the parameters used in the component cost function Equations (7.4) are in Table 7.11. Other parameters used in the physical programming are given in Table 7.12. The utility constraint value, the cost constraint value and the reliability constraint value are chosen as $C_0 = 15$, $U_0 = 1$ and $R_0 = 0.5$.

Table 7.11: Parameters used in the component cost function

$g_1 = g_2$	$f_1 = f_2$	$U^l_{1\_} = U^l_{2\_}$	$P_{1\_} = P_{2\_}$	$P_{1\_} = P_{2\_}$
1	0.99	1	0.93	0.9999

Table 7.12: Physical programming class functions setting

	g_1	g_2	g_3	g_4	g_5
Utility	2.5	2	1.8	1.5	1
Cost	1	5	7	10	15
Reliability (Probability of the system in state 2 or above)	1.0	0.8	0.7	0.6	0.5

After developing the objective functions and their constraints as shown in Equation (7.7) to (7.10), we used the GA toolbox (Houck et al., [10]) in Matlab R2006a to solve the problems. We used a computer with 2.00GHZ AMD Opteron (tm) Processor and 3.00 GB RAM under Windows XP operating system to run these programs. The population data type has been set to be double and population size to be 20. Elite count and crossover fraction are all set to be the default values in Matlab R2006a GA toolbox. The uniform function was used as the population creation function. The adoptive feasible function was used as the mutation function. The rank function was used as the scaling function. The scattered function was used as the crossover function. The stochastic uniform function was used as the selection function. Stopping criteria are set as 1000 generations, 500 stall generations, 100 stall time limit and 10^{-6} function tolerance. The reason we set the stopping criteria like this is that we found the results can be convergent for all the calculations. There are 28 decision variables in each of the above optimization problem. These problems are all non-linear non-integer optimization problems. It is hard to use traditional optimization approaches to solve these problems. However, by using GA, they were solved within a few minutes.

Table 7.13: Comparison of the single-objective optimization results and multi-objective optimization results for multi-state weighted serials-parallel

	Single-objective optimization results Minimize Cost				Single-objective optimization results Maximize Reliability				Single-objective optimization results Maximize Utility				Multi-objective optimization results			
Reliability	0.5000				0.9946				0.6236				0.8538			
Utility	1.5766				2.0355				2.2283				2.0626			
Cost	10.2824				15.0000				15.0000				11.9884			
Reliability distribution	0.0079	0.4921	0.5000	0.0000	0.0025	0.0029	0.9946	0.0000	0.0087	0.3677	0.5271	0.0965	0.004	0.1422	0.8538	0.0000
U	0.0000	2.1599	1.0205	1.1192	0.0000	2.0048	2.1948	2.0428	0.0689	0.3102	0.3105	0.3104	0.0666	0.3116	0.3110	0.3108
	0.0000	2.0000	2.0076	1.2393	0.0000	2.0542	2.0019	2.0000	0.0550	0.1720	0.1720	0.6010	0.0001	0.5102	0.2449	0.2448
	0.0000	1.0049	2.3147	1.0326	0.0000	2.8870	2.0114	2.0000	0.0667	0.3111	0.3111	0.3111	0.0682	0.3076	0.3093	0.3149
	0.0000	1.1465	2.0162	2.0000	0.0000	2.0647	1.0374	2.0066	0.0689	0.3101	0.3102	0.3108	0.0689	0.1445	0.6419	0.1447
P	0.0700	0.3101	0.3099	0.3100	0.0398	0.3856	0.2887	0.2859	0.0000	1.0052	1.8284	3.0036	0.0000	2.5001	1.0694	1.1391
	0.0700	0.3100	0.3100	0.3100	0.0509	0.3164	0.3163	0.3164	0.0000	1.1569	2.7357	2.5000	0.0000	2.0147	2.1277	2.1435
	0.0546	0.4691	0.1326	0.3437	0.0081	0.3307	0.3307	0.3305	0.0000	2.5796	2.9613	1.4754	0.0000	1.0578	1.0625	2.6498
	0.0552	0.3152	0.3152	0.3144	0.0407	0.2312	0.3203	0.4078	0.0000	1.0162	3.0001	1.8236	0.0000	1.7821	2.1406	2.8452

In Table 7.13, the reliability distribution is the probability of the system in each state. We can observe that even the components have four states, the system may only have three states or even two states (binary system). In the single-objective optimization problem which the system cost is used as the objective, the optimal system is a three-state system. It makes sense. Because a three-state system is more cost effective than a four-state system with the same requirements in system utility and the probability of the system in state 2 or above. In the single-objective optimization problem which the probability of the system in state 2 or above is used as the objective, the optimal system is almost a two-state system (the probability in state 0 and state 1 are too less to compare with the probability in state 2). First, a three-state system is more cost effective than a four-state system with the same requirements in the probability of the system in state 2 or above, so

the probability in state 3 is 0. Second, the cost limitation is enough to put a much higher probability in state 2 than in state 0 and state 1 when the objective is to maximize the probability of the system in state 2 or above. So the result comes up with an almost binary system.

From the optimization procedures and results, we can see that the physical programming method has advantages over the single-objective optimization method. A problem of using single-objective optimization approach is that maybe the resulting optimal reliability and utility are good while the cost objective is totally unsatisfying. Of course we can adjust the constraint functions when encountering this problem, but we can not ensure that the optimal results will be satisfactory next time. When using the physical programming approach, however, with class functions accurately describing the decision maker's preferences on each objective and with the One vs. Others rule as the inter-criteria preference, we can ensure to get satisfactory optimal results with respect to each design criterion. The class function provides the preference inside a single objective, and the One vs. Others rule of physical programming provides the preferences among different objectives. Therefore, physical programming will drive the optimal values of different objectives to preference ranges close to one another.

7.5 An alternative model of the system

Based on different application situations, an alternative model of MS weighted series-parallel system is presented here:

> The MS weighted series-parallel system is in state j or above if in every parallel subsystem, the utility of at least one component in state j or above is equal to or

bigger than the given value k_j , which also means every subsystem has to be in state j or above.

This model will be referred to as the alternative model, while the one given in Section 2.1 will be referred to as the original model in the following discussion. The two types of MS weighted series model are based on different practical application situations. In the original model, we do not care about the state (the health situation) of the component and only consider the utility of the component. In the alternative model, we not only care about the utility of the component but also the state (the health situation) of the component. Sometime even the component has the utility as we need, because its health situation is not good enough which is symbolized by its state, it can not meet the system requirement. Let's see an example.

Example 7.5: There are three cities: A, B and C. Two fiber optic cables connect city A and city B. Another two fiber optic cables connect city B and city C. Each cable can work under three states: fail, marginal and perfect. The bandwidth of each cable is its utility. The utility distribution of each cable is the same as in Example 7.3. If we require each cable provide at least 5 Mbps bandwidth whatever the state it is in, the original model is suitable to be applied. If we require each cable provide at least 5 Mbps bandwidth and must work in the perfect state, the alternative model should be used.

For the same reason as mentioned before, a weighted average utility value should be used to describe the system utility in each state. The calculation process is similar as in the original model. The only thing needed to be kept in mind is that in the alternative model, only the component in state j or above can contribute to the system utility in state j.

After the weighted average utility value in each state has been calculated, Equation (7.6) can be used again to calculate the system utility.

In the alternative model, in order to let the system be in state j or above, in each parallel subsystem at least the utility of one component in state j or above is equal to or bigger than the given value k_j. The probability of the ith parallel subsystem in state j or above is:

$$\begin{aligned}&\Pr\left(\phi_i(x_i) \geq j\right) = \Pr\left(\gamma_i(u_i) \geq k_j\right) \\ &= \Pr\left\{\left(u_{i1}^j \geq k_j\right) \cup \left(u_{i2}^j \geq k_j\right) \cdots \cup \left(u_{ir}^j \geq k_j\right) \cup \cdots \cup \left(u_{in_i}^j \geq k_j\right)\right\}\end{aligned} \tag{7.11}$$

where n_i is the number of components in the ith parallel subsystem and u_{ir}^j is the utility of the rth component in the ith parallel subsystem when that component's state is equal to or greater than j. According to the above analysis, in a similar way as in the original model, the alternative MS weighted series-parallel system model can also be decomposed into a binary series-parallel system. And then the binary series-parallel system reliability evaluation methods can be applied to evaluate the system reliability.

For the alternative model, another component cost function developed in Li and Zuo [14] and [18] can be used to describe the relationship between component cost, reliability and utility:

$$c_i^j = g_i^j \cdot \exp(U_i^j - U_{i_{min}}^j) + \exp[(1 - f_i^j) \cdot \frac{P_i^j - P_{i_{min}}^j}{P_{i_{max}}^j - P_i^j}] \tag{7.12}$$

where $U_i^j = \sum_{r=j}^{M} p_{i,r} * u_{i,r}$ and $P_i^j = \sum_{r=j}^{M} p_{i,r}$

- $P_{i_{min}}^{j}$: minimum probability of component i being in the state j or above
- $P_{i_{max}}^{j}$: maximum probability of components i being in the state j or above
- f_i^j : feasibility of increasing the probability of component i being in the state j or above
- $U_{i_{min}}^{j}$: minimum value for the weighted average utility of component i being in the state j or above
- g_i^j : feasibility of increasing the weighted average utility of component i being in the state j or above
- c_i^j : cost of component i for the alternative model if the component can contribute to the system in the state j or above
- P_i^j : the probability of component i being in state j or above

Actually Equation (7.12) is just Equation (6.7) without superscript II. As we have mentioned in chapter 6, the difference between the original model and the alternative model is whether the components whose states are below j are making any contribution for the system to be in state j or above. In the alternative model, only the components whose states are in state j or above can make contributions for the system to be in state j or above. It is the reason why we have $U_i^j = \sum_{r=j}^{M} p_{i,r} * u_{i,r}$ and $P_i^j = \sum_{r=j}^{M} p_{i,r}$. The physical meaning of parameters in Equation (7.12) is similar like in Equation (7.4). Equation (7.12) is to calculate the cost of each component. The total

investment cost can be calculated by adding the cost of each component together just as shown in Equation (7.5).

Table 7.14: Comparison of the single-objective optimization results and multi-objective optimization results for the alternative multi-state weighted serials-parallel system

	Single-objective optimization results Minimize Cost				Single-objective optimization results Maximize Reliability				Single-objective optimization results Maximize Utility				Multi-objective optimization results			
Reliability	0.5000				0.9971				0.9328				0.9005			
Utility	1.5055				2.0923				2.2303				2.1147			
Cost	10.0007				15.0000				15.0000				12.7993			
Reliability distribution	0.0166	0.4834	0.5000	0.0000	0.0006	0.0023	0.9971	0.0000	0.0090	0.0582	0.9328	0.0000	0.0088	0.0943	0.9005	0.0000
U	0.0000	0.0000	1.0084	2.0004	0.0000	0.0000	2.1765	2.2186	0.0000	0.0000	1.7109	2.4698	0.0000	0.0000	2.1117	2.2493
	0.0000	0.0000	1.0018	2.0004	0.0000	0.0000	2.0051	2.4756	0.0000	0.0000	2.3696	2.1504	0.0000	0.0000	2.2706	1.1068
	0.0000	0.0000	2.0042	1.0083	0.0000	0.0000	2.0152	2.0115	0.0000	0.0000	2.2149	2.2237	0.0000	0.0000	2.2266	2.2501
	0.0000	0.0000	2.0058	1.0351	0.0000	0.0000	1.3663	2.1777	0.0000	0.0000	2.4714	1.6142	0.0000	0.0000	1.4695	1.6054
P	0.0284	0.0335	0.4688	0.4683	0.0028	0.0036	0.4969	0.4969	0.0349	0.0335	0.4674	0.4652	0.0270	0.0345	0.4738	0.4657
	0.0349	0.0356	0.4651	0.4649	0.0318	0.0338	0.4673	0.4678	0.0338	0.0335	0.4669	0.4668	0.0335	0.0338	0.4156	0.5182
	0.0322	0.0331	0.4664	0.4674	0.0013	0.0036	0.4981	0.4974	0.0276	0.0350	0.4732	0.4651	0.0348	0.0349	0.4654	0.4659
	0.0325	0.0348	0.4584	0.4733	0.0222	0.0118	0.4780	0.4875	0.0347	0.0351	0.4653	0.4660	0.0335	0.0335	0.4670	0.4670

With the functions of system reliability, expected system utility and total investment cost for the alternative model, we can formulate the same single-objective optimal design models and the multi-objective model as in Section 3 (Equation (7.7) to (7.10)). Using the same data and approach as in Section 4, the single-objective optimization results and multi-objective optimization results are compared for the alternative model and illustrated in Table 7.14. Clearly, from Table 7.14 we can obtain the same observations as from Table 7.13 about the advantages of the multi-objective optimization approach.

7.6 Conclusions

This chapter brought forward two models for the multi-state weighted series-parallel system and provided the approach to calculate system reliability and the expected system utility. This chapter also proposed a multi-objective optimization model for multi-state weighted series-parallel system optimal design. This model seeks to maximize system reliability and performance utility while minimizing system cost simultaneously. Physical programming is used as an effective approach to build the optimal model within this multi-objective optimization framework. Genetic algorithm is used in solving the proposed physical programming based optimization models. Finally, an example for each multi-state weighted series-parallel model has been given to illustrate the flexibility and effectiveness of the proposed physical programming over the single-objective method.

From Chapter 4 to Chapter 7, we introduced three reliability system models: binary weighted k-out-of-n system, MS weighted k-out-of-n system and MS weighted series-parallel system. The definitions and reliability evaluation methods have been described for each model in details. Different reliability optimal design problems have been studied for these models. For the binary weighted k-out-of-n system, the most commonly studied reliability optimal design problem – component selection problem has been studied. For the MS weighted k-out-of-n system, we brought forward and solved the new more general reliability optimal design problem – component design problem. For the MS weighted series-parallel system, we not only studied the component design problem similar as for MS weighted k-out-of-n system, but also studied the multi-objective optimization model which can provide more reasonable optimal solutions. In the next

chapter, we will summarize all the research in this thesis and discuss the directions in the future research.

References:

[1]. M. Agarwal and R. Gupta. Homogeneous redundancy optimization in multi-state series-parallel systems: A heuristic approach. IIE Transactions, 39 (3): 277-289, 2007.

[2]. T. Aven. On performance-measures for multi-state monotone systems. Reliability Engineering & System Safety, 41 (3): 259-266, 1993.

[3]. R.E. Barlow and A.S. Wu. Coherent systems with multi-state components. Mathematics of Operations Research, 3 (4): 275-281, 1978.

[4]. H.W. Block and T.H. Savits. A decomposition for multi-state monotone systems. Journal Applied Probability, 19: 703-714, 1982.

[5]. P.A. Wood. Multi-state block diagrams and fault trees. IEEE Transactions on Reliability, R-34 (3) : 236-240, 1985.

[6]. B. Dodson and D. Nolan. Reliability Engineering Handbook. Marcel Dekker Inc, 1999.

[7]. D.W. Coit. Maximization of system reliability with a choice of redundancy strategies. IIE Transactions, 35 (6): 535-543, 2003.

[8]. D. Kececioglu. Reliability Engineering Handbook. Volume 2, PTR Prentice Hall, 1991.

[9]. R. Gupta and M. Agarwal. Penalty guided genetic search for redundancy optimization in multi-state series-parallel power system. Journal of Combinational Optimization, 12 (3): 257-277, 2006.

[10]. C. Houck, J. Joines and M. Kay. A Genetic Algorithm for Function Optimization: A Matlab Implementation. North Carolina State University - Industrial Engineering Technical Report 95-09, 1995.

[11]. W. Kuo and V.R. Prasad. An annotated overview of system-reliability optimization. IEEE Transactions on Reliably, 49 (2): 176-187, 2000.

[12]. L.M. Leemis. Reliability, Probabilistic Models and Statistical Methods. Rentice-Hall, 1995.

[13]. W. Li and M.J. Zuo. Optimal design of binary weighted *k*-out-of-*n* systems. Proceedings of the 2006 Industrial Engineering Research Conference, Orlando, U.S.A, 2006.

[14]. W. Li and M.J. Zuo. Optimal design of multi-state weighted *k*-out-of-*n* systems based on component design. Proceedings of European Safety and Reliability Conference, Stavanger (Norway), 25-27 June, 2007.

[15]. W. Li and M.J. Zuo. Optimal Design of multi-state weighted series-parallel systems using physical programming and genetic algorithms. Proceedings of the 7th International Conference on Reliability, Maintainability and Safety, Beijing, China, 2007.

[16]. W. Li and M.J. Zuo. Reliability evaluation of multi-state weighted *k*-out-of-*n* systems. Reliability Engineering and System Safety, 93 (1): 161-168, 2008.

[17]. W. Li and M.J. Zuo. Optimal Design of multi-state weighted series-parallel systems using physical programming and genetic algorithms. Submitted to IEEE Transaction on Reliability, 2008.

[18]. W. Li and M.J. Zuo. Optimal design of multi-state weighted *k*-out-of-*n* systems based on component design. Reliability Engineering & System Safety, Accepted for Publication (available online), 2008.

[19]. A. Lisnianski and G. Levitin. Multi-state System Reliability: Assessment, Optimization and Applications. World Scientific, Singapore, 2003.

[20]. P.X. Liu, M.J. Zuo and M. Q-H Meng. A neural network approach to optimal design of continuous-state parallel-series systems. Computers and Operations Research, 30: 339-352, 2003.

[21]. Y. Massim, R. Meziane, A. Zeblah and M. Rahli. Ant colony optimization for multi-state series-parallel system expansion scheduling. Electrical Engineering, 87 (6): 327-336, 2005.

[22]. Y. Massim, A. Zeblah, R. Meziane, M. Benguediab and A. Ghouraf. Optimal design and reliability evaluation of multi-state series-parallel power systems. Nonlinear Dynamics, 40 (4): 309-321, 2005.

[23]. R. Meziane, Y. Massim, A. Zeblah, A. Ghoraf and R. Rahli. Reliability optimization using ant colony algorithm under performance and cost constraints. Electric Power Systems Research, 76 (1-3): 1-8, 2005. .

[24]. A. Messac. Physical programming: effective optimization for computational design. AIAA Journal, 34 (1): 149-158, 1996.

[25]. A Messac and A. Ismail-Yahaya. Multi-objective robust design using physical programming. Structural and Multidisciplinary Optimization, J. of the Int. Society of Structural and Multidisciplinary Optimization (ISSMO), 23 (5): 357-371, 2002.

[26]. A. Messac, M. Martinez and T. Simpson. Introduction of a product family penalty function using physical programming. ASME J. of Mechanical Design, 124 (2): 164-172, 2002.

[27]. A. Messac and B. Wilson. Physical Programming for Computational Control. AIAA Journal, 36 (2): 219-226, 1998.

[28]. A. Mettas. ReliaSoft Corporation and Tucson, Reliability allocation and optimization for complex systems. Proceedings of Annual Reliability and Maintainability Symposium, Los Angeles, California, USA, January 24-27, 2000.

[29]. M. Nourelfath and D. Ait-Kadi. Optimization of series-parallel multi-state systems under maintenance policies. Reliability Engineering and System Safety, 92 (12): 1620-1626, 2007.

[30]. M. Patel, K.E. Lewis, A. Maria and A. Messac. System design through subsystem selection using physical programming. AIAA Journal, 41 (6): 1089-1096, 2003.

[31]. J.E. Ramirez-Marquez and D.W. Coit. A heuristic for solving the redundancy allocation problem for multi-state series-parallel. Systems Reliability Engineering and System Safety, 83 (3): 341-349, 2004.

[32]. K. Shen and M. Xie. On the increase of system reliability by parallel redundancy. IEEE Transactions on Reliability, 39 (5): 607-611, 1990.

[33]. Z. Tian, M.J. Zuo and H. Huang. Reliability-redundancy allocation for multi-state series-parallel systems. Proceedings of the 2005 European Safety & Reliability Conference (ESREL05), Tri City, Poland, pp. 1925-1930, 2005.

[34]. Z. Tian and M.J. Zuo. Redundancy allocation for multi-state systems using physical programming and genetic algorithms. Reliability Engineering and System Safety, Special issue on GA in Reliability, 91 (9): 1049-1056, 2006.

Chapter 8
Conclusions and Future Work

Reliability is one of the most critical performance measures of today's complex systems, one emphasized more and more by academia, industry and government. The reliability of a system needs to be evaluated accurately; this can be achieved through optimal reliability design.

8.1 Conclusions

Traditional reliability theory simplified all the real-world systems into a binary model — both systems and components can take only two possible states: completely working and totally failed. However, engineering systems typically have multiple partial failure states in addition to the above-mentioned completely working and totally failed states. Multi-state reliability theory recognizes the multiple possible states of engineering systems. It can map the performance of components to system performance more accurately, and be able to provide better solutions through the optimal reliability design. In this thesis, we brought forward a new MS system model — the MS weighted system model. The research on MS weighted system reliability modeling, evaluation and optimal design makes significant contributions to MS system reliability theory, and has potential applications in the oil & gas production industry, manufacturing industry, power industry and so on. The main contributions of this thesis are summarized below.

1. **An efficient algorithm for the reliability evaluation of MS weighted *k*-out-of-*n* systems**

We have proposed two definitions for the multi-state weighted *k*-out-of-*n* system model. They may be applied in different situations when the contributions of relatively weak components may or may not be useful. Recursive algorithms are developed for evaluation of system distribution under both definitions. The UGF approach is compared with recursive methods for the binary weighted *k*-out-of-*n* system defined by Wu and Chen [1] and the proposed two types of multi-state weighted *k*-out-of-*n* systems. It is found that recursive methods are generally more efficient than the UGF approach.

2. **An algorithm for the reliability evaluation of MS weighted series-parallel systems**

Based on a different logical structure from the MS weighted *k*-out-of-*n* system model, we brought forward two types of MS weighted series-parallel system models. As a coherent system, the MS weighted series-parallel system can be decomposed into the traditional binary system. Then, traditional binary system reliability evaluation methods can be used to evaluate the MS weighted series-parallel system reliability. We developed the decomposing approach based on the MS weighted series-parallel system construction function.

3. **Reliability optimal design based on component selection for binary weighted *k*-out-of-*n* systems**

Although the component selection optimal design problem has been widely studied, in this thesis, it is studied for the binary weighted *k*-out-of-*n* system for the first time,

and two optimal models are formulated. One minimizes the expected total cost while guaranteeing a system reliability greater than a pre-specified value; the other maximizes system reliability with the constraints on total system cost. Genetic Algorithm (GA) and Tabu Search (TS) methods are both used to solve the resulting optimization models. The results show that both are powerful tools for solving these kinds of problems, but TS is more efficient.

4. **Reliability optimal design based on component design for MS weighted *k*-out-of-*n* systems**

   Although the "component selection problem", which involves selection of components with known reliability and cost characteristics has been widely studied, less adequately addressed has been the more general problem of determining system cost and utility based on the relationship between component reliability, cost and utility. In this thesis, this optimal design has been addressed for the first time. Two optimal design problems have been studied for MS weighted *k*-out-of-*n* systems in this thesis, one is to minimize the expected total system cost subject to system reliability requirements, the other is to maximize system reliability subject to total system cost limitations. The resulting optimization problems are too complicated to be solved by traditional optimization approaches, therefore Genetic Algorithm (GA) is used to solve them. Our results show that GA is a powerful tool for solving these kinds of problems.

5. **Multi-objective optimal design for MS series-parallel systems based on component design**

In the general single-objective optimization models, there exist mutually conflicting goals such as system reliability and total system cost. We treat one goal as the objective function of the optimization model and the other as the constraint. It is very difficult to specify in advance the value that should be given to the goal that acts as a constraint. After a solution is obtained, we often need to modify the constraint value to find a better trade-off between goals such as system reliability and system cost. Finding the most appropriate constraint value is a trial and error process that can be time-consuming. In contrast to the traditional single-objective optimization model, the optimization model proposed here is a multi-objective optimization model which is used to maximize expected system performance utility and system reliability while simultaneously minimizing total investment cost. Physical programming technique is used to build these multi-objective optimization models. Genetic algorithm is used to solve the proposed physical programming based optimization model. The flexibility and effectiveness of the proposed multi-objective models are confirmed through comparison with single-objective models with regard to optimal solutions.

With these efficient reliability evaluation methods and effective reliability-based design approaches for multi-state engineering systems, the research results will provide useful tools for achieving highly reliable and cost effective engineering systems.

8.2 Future work

In the future, research can be focused on investigations along the following directions and applying the research results to practice:

1. **Multi-State Weighted Component Criticality and Importance Analysis:**

In general, importance measures are used to evaluate and rank the criticality of component or component states with respect to system reliability. The focus of my study will be to provide intuitive and clear importance measures that can be used to direct our reliability design work from two perspectives: (1) how a specific component affects multi-state weighted system reliability and (2) how a particular component state or set of states affects multi-state weighted system reliability. In addition, the multi-state weighted redundancy importance will be studied to identify where to allocate component redundancy in order to improve system reliability.

2. **Evaluation and Fault-Tolerant Design of Multi-State Weighted Network Systems:**

Many real-world systems are multi-state weighted networks; among these are computer and communication systems, power transmission and distribution systems, transportation systems, and oil/gas production systems. Evaluating network reliability is an important topic with regard to the planning, designing, and control of systems. However, at present, most of the research in the literature still simply considers the network as a binary system. In the future, I will extend the research to the reliability evaluation and fault-tolerant design of network systems based on the multi-state weighted model, which refers to system performance evaluation based on continuous degradation and design of a network so it will continue to operate, possibly at a reduced level at they undergo what is known as "graceful degradation".

3. Condition Based Maintenance Systems Based on Multi-State Weighted Reliability Theory:

Maintaining and repairing industrial facilities constitutes a significant portion of many companies' annual operating budgets. Although the idea of CBM (Condition Based Maintenance) which uses the facility health condition data to make maintenance decisions is not new, utilization of new technologies such as degradation reliability models and multi-state reliability theory in CBM is a new direction of research. The objective of my research is to develop an innovative and effective CBM decision making system by combining my multi-state weighted research results with remaining life prediction and maintenance scheduling optimization techniques.

Based on the research I have done for this thesis, this proposed research will assuredly make significant contributions to both academia and industry.

References:

[1]. J. S. Wu and R. J. Chen. An algorithm for computing the reliability of a weighted k-out-of-n system. IEEE Transactions on Reliability, R-43(2): 327-328, 1994.